ENCICLOPEDIA PRACTICA DE LA

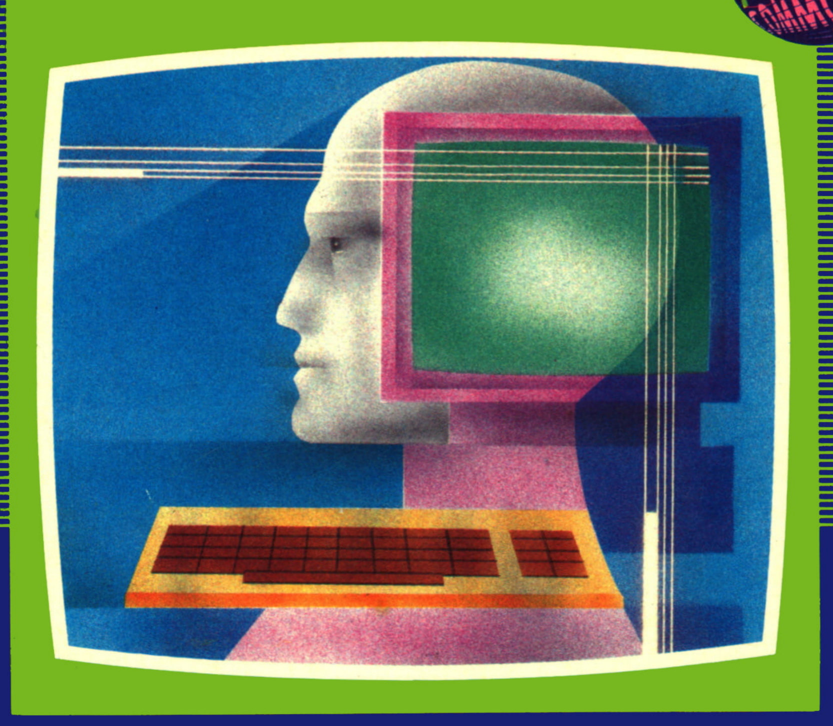
INFORMATICA APLICADA

40

¿Máquinas más expertas que los hombres?

José Arteche

SINCLAIR
AMSTRAD
COMMODORE
IBM



EDICIONES SIGLO CULTURAL

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

40

¿Máquinas más expertas que los hombres?

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:
BRAVO-LOFISH.

Dibujos:
JOSE OCHOA Y ANTONIO PERERA.

Tomo XXXX. **¿Máquinas más expertas que los hombres?**
JOSE ARTECHE, Ingeniero de Telecomunicación

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta (Ed. Iberia Mart I). Teléf. 810 52 13. 28020 Madrid

Publicidad:
Gofar Publicidad, S.A. San Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-162-2
ISBN de la obra: 84-7688-018-9.

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 23.539-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta (Ed. Iberia Mart I). Teléf. 810 52 13. 28020 Madrid
Agosto, 1987
P.V.P. Canarias: 365,-

INDICE



A Blanca

INTRODUCCION 1



QUEREMOS comentar en esta introducción cual ha sido nuestro objetivo en la preparación de este libro y qué va a encontrar el lector en él.

Siempre es comprometido explicar qué se ha pretendido al elaborar cualquier trabajo (todo el que ha escrito un libro lo sabe bien) porque se corre el riesgo de parecer excesivamente modesto en las aspiraciones o, en el otro extremo, sumamente pretencioso. Sin embargo, hemos creido de utilidad para el lector hacer esta aclaración, por lo que la incluimos aquí.

Nuestra intención es que este libro sea una introducción a los «sistemas expertos» (SE) y sirva de guia en este fascinante área de la Inteligencia Artificial.

Intentamos que el lector medio de esta colección (supuestamente no experto en la materia) adquiera un conocimiento, necesariamente no muy profundo, pero claro y suficiente sobre:

a) qué significan los sistemas «basados en el conocimiento» dentro de la Informática;
b) qué relación guardan con otras disciplinas próximas dentro del amplio mundo de la Inteligencia Artificial;
c) para resolver qué problemas pueden ser utilizados o qué aportan en este sentido, y
d) hasta qué punto son útiles, en qué medida son «expertos» y hasta donde pueden suplir o complementar las actividades de un experto humano.

El segundo capítulo aborda estos temas y pretende centrar las espectativas depositadas en esta supuesta «panacea universal» de la informática, en sus justas proporciones. Los SE son un instrumento informático muy util en numerosas ocasiones pero «no lo resuelven todo». Es interesanteu-

til saber qué puede hacerse con un SE y qué no; en qué es semejante a otros tipos de aplicaciones informáticas y en qué difiere de ellas.

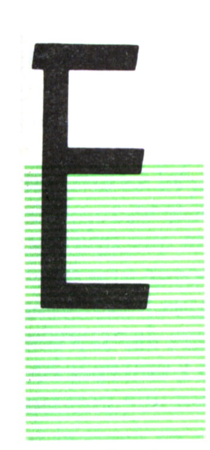
En el capítulo tercero se describe en detalle la estructura interna típica de un SE, pero mas con la intención de aclarar el alcance de este tipo de sistemas que como guia «construya su propio sistema experto». En efecto, entendemos que la implementación de un sistema informático minimamente válido es mas bien tarea de profesionales experimentados. Si cualquier persona interesada en este tema desea desarrollar un SE nos atrevemos a sugerirle que implemente dicho sistema utilizando alguna de las «herramientas» que existen en el mercado (algunas disponibles hasta para un PC). En el capítulo cuatro ofrecemos información al respecto.

De todos modos, existen libros interesantes sobre como construir un sistema experto «casero», y sus referencias aparecen en la Bibliografia.

En resumen, entendemos que la lectura de estas páginas puede ser util a dos tipos de lectores (entre otros): a quien quiera tener una idea clara de qué es y para que sirve un SE (bien profesionales de la Informática expertos en otras areas, bien personas con menos conocimientos pero interesados en los aspectos mas novedosos de esta ciencia) y a quienes deseen llegar a implementar un SE, como introducción a su estructura y características y como guia en la selección de las «herramientas» adecuadas a la tarea que pretenden acometer.

Esperamos, en todo caso, que cualquier persona que comience la lectura de este libro no se sienta defraudada sino que tenga una lectura amena y lo concluya con provecho.

LOS «EXPERTOS NO HUMANOS» 2



EL título de este libro trata de plasmar una de las preguntas mas usuales que suelen hacerse las personas interesadas por los temas de Informática pero no muy introducidos a nivel profesional en los desarrollos recientes en Inteligencia Artificial. Se habla de «sistemas expertos» y de «sistemas inteligentes», pero ¿hasta que punto pueden ser «inteligentes» estos sistemas? Se comenta que los «sistemas expertos» sustituyen (o pueden sustituir) a los «expertos humanos» en cantidad de tareas, pero ¿es posible que una máquina o un sistema informático sea más experto, o al menos «tan» experto como el experto humano? Pretendemos contestar a estas preguntas de un modo sencillo y claro.

Los sistemas expertos son una aplicación concreta de las muchas que se desarrollan dentro del área informática de la Inteligencia Artificial. Son la aplicación más popular hoy en día y la más extendida.

Un «sistema experto» es un sistema informático que contiene informaciones diversas sobre un área de actuación humana y puede obtener conclusiones sobre esos datos que se le han introducido, sugerir modos de actuación, tomar decisiones, etc.

En cierto modo, todas estas actividades pueden ser desarrolladas (al menos en parte, o con limitaciones) por los sistemas informáticos convencionales. Sin embargo, en los «sistemas expertos» se introducen un conjunto de técnicas propias del enfoque de la I.A. y un punto de vista diferente (como veremos más adelante) que permiten un aprovechamiento mayor y mejor de los datos de que se dispone.

Básicamente es este punto de vista (más próximo al del pensamiento humano) el que ha potenciado el desarrollo de unas técnicas específicas de tratamiento de los datos (en Inteligencia Artificial se habla más bién de «conocimientos») y de su representación para permitir este otro modo de trabajo.

Y ¿en qué consiste este punto de vista «más proximo al del pensamiento humano»? Está formado por un conjunto de elementos y técnicas muy específicas que los técnicos aplican, pero que podríamos sintetizar diciendo que se trata de hacer que los datos (informaciones, conocimientos) se almacenen como los piensa una persona (como los conocen los «expertos») y que la máquina se adecue al modo de razonar de estos expertos sobre las informaciones que ellos manejan, en vez de hacer que los expertos (o los profesionales de la informática que les asesoran) creen algoritmos de proceso y estructuras de datos adecuados a la organización y modo de trabajo de las máquinas.

Se trata de procesar informaciones del tipo «SI sobre un punto de la tierra la presión atmosférica es de 1024 mm y la zona está sometida a vientos de componente NW de 20 nudos y ... y, además, en los últimos diez años cuando se han dado estas circunstancias en un 70% de los casos ha evolucionado la situación hacia ... y, además, las estimaciones del Centro de Pronósticos preveían lluvias abundantes para el final de este mes, y ... ENTONCES debe suceder que ...». En un enfoque clásico, habría que tipificar y clasificar estas situaciones, asignarles un código y establecer una base de datos con la situación actual y otra (o la misma) con datos históricos; después habría que definir un algoritmo (construir un proceso de acuerdo con las fórmulas adecuadas) para el tratamiento de los datos y entregar los datos resultantes al experto para su análisis y para que hiciera una presentación «inteligible» de las predicciones.

Naturalmente, hay numerosos procesos en los que los datos ya están cuantificados y el tratamiento a efectuar está perfectamente definido en un algoritmo claramente formulado (piénsese en la preparación de una nómina o en la emisión de una factura) en los que, por tanto, este enfoque que comentamos no aportaria nada nuevo. Hay, sin embargo, numerosas situaciones en que es casi imprescindible un planteamiento más flexible. De hecho, sólo mediante la puesta en marcha de sistemas expertos se están encontrando soluciones satisfactorias (y no en la opinión de todos los profesionales implicados) para problemas como el diagnóstico médico o la previsión de averias en sistemas electrónicos complejos.

El sistema experto, como veremos, consta básicamente de un conjunto de «informaciones» sobre el problema de que se trata («base de conocimiento») y un programa que obtiene resultados a partir de esos conocimientos («motor de inferencia»). En la base de conocimiento se incluyen elementos muy diversos: desde simples hechos que se constatan («Ahora la presión atmosférica es de 1024 mm») hasta reglas sobre cómo deben procesarse los datos básicos («Si la presión atmosférica es superior a la normal y la situación de la borrasca es al NW, entonces es probable —probabilidad del 70%— que llueva»), e incluso normas sobre cómo deben ser utilizadas las reglas de la propia base de conocimientos («Si en los últimos 10 días no ha habido precipitaciones apreciables, pero la presión atmosfé-

rica se ha mantenido por debajo de 1000 mm, entonces tomar en consideración la serie histórica de datos de los últimos 5 años»). A su vez el motor de inferencia es un programa que procesa los datos de la base de conocimiento de un modo inteligente.

Aparte el hecho de que los conocimientos se representan según esquemas muy especiales en un sistema experto (casi las frases anteriores podrían ser directamente elementos de la base de conocimientos, tal como están escritas), además el motor de inferencia aporta unos paradigmas de búsqueda de datos y obtención de conclusiones muy interesantes (muy eficaces) y aporta elementos muy peculiares como el hecho de estar preparado para preguntar algún dato si «le falta» para poder completar un proceso de deducción, el que admita que se le controle (por parte del usuario) en su proceso de deducción, etc.

Profundizaremos en esta descripción de los sistemas expertos y su estructura en el próximo capítulo; mientras veremos ahora el significado de este tipo de sistema informático, sus aportaciones prácticas y sus áreas de aplicación.

Históricamente, los primeros sistemas de ordenadores (o máquinas automáticas de cálculo) se utilizaron para sencillos procesos de cálculo (poco mas que las cuatro operaciones aritméticas básicas y clasificaciones); posteriormente, a medida que la capacidad de proceso de los ordenadores aumentaba, se les fueron encomendando tareas mas complicadas, siempre en la línea de más cantidad de proceso, sin cambiar de modo de trabajo. Se llegó a pensar, hace unos años, que el ordenador «podría con todo». Posteriormente se ha ido evolucionando en la línea de modificar el modo de (pensar» u operar del ordenador para poder abordar «con más astucia» y no sólo con «más fuerza» problemas complejos de procesamiento de informaciones. De hecho, lo que se anuncia como quinta generación de ordenadores (y así se llama el proyecto japonés que está examinando este proceso de desarrollo) sigue la línea de evolución que comentamos y es que la que se utiliza en los sistemas expertos. Independientemente del éxito que este proyecto pueda tener y de la fecha de disponibilidad de realizaciones concretas (temas ambos muy debatidos hoy en día) parece claro que la informática de los años 2000 pasará por este tipo de enfoques y utilizará este tipo de técnicas.

Parece que, en general, es adecuado un enfoque como el que se utiliza en los sistemas expertos cuando no se conoce un método concreto (standard) para interpretar (procesar) los datos: no se puede llegar a definir un algoritmo preciso, sino que los razonamientos de las personas que utilizan esos datos son imprecisos o inconcretos; e, incluso, el modo de razonar puede diferir de unos expertos a otros, aunque los resultados a los que lleguen sean equivalentes (o, normalmente es así). Un caso típico de imprecisión de la información se produce, por ejemplo, en la interpretación de textos: una palabra puede tener un significado u otro dependiendo del con-

texto en que se encuentre; o bien en el reconocimiento de signos gráficos: un signo como el que aparece en la figura puede ser interpretado como una letra «n» o como una «m» o como una «u» dependiendo del resto de la palabra a la que pertenece.

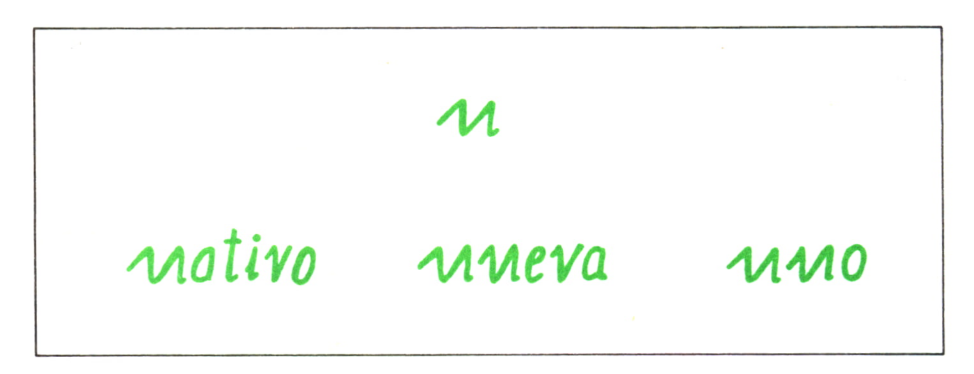


Fig. 1. Interpretación de un signo gráfico según la palabra a la que pertenece.

En el diagnóstico médico o en la predicción meteorológica un síntoma determinado será apreciado de un modo diverso según el resto de síntomas presentes y la fiabilidad de las conclusiones dependerá del número y tipo de los síntomas o indicios de los que hemos partido para llegar a esa conclusión.

Otro caso, sencillo pero ilustrativo, de cómo en ocasiones el «procesamiento de datos» humano difiere del que realiza la máquina se da cuando realizamos la suma de dos cifras: el ordenador siempre la realizará (groso modo) almacenando cada dato en un registro y sumando los digitos de derecha a izquierda calculando los «arrastres» correspondientes (bien hasta el final del campo previsto para almacenamiento de los datos, aunque el final, por la izquierda, sean sólo ceros no significativos; bien parando el cómputo cuando se ha terminado uno de los dos sumandos, si están contenidos en campos de distinta longitud). Por el contrario una persona examinará ante todo el tamaño de las cifras a sumar: si son de un solo dígito (o, a lo sumo, una de ellas de dos dígitos) realizará «la cuenta» mentalmente de un modo global (5+7=12; 37+8=45), si una de ellas es de tres o mas dígitos y la otra de uno solo, realizará la operación sumando mentalmente los dos últimos dígitos (unidades y decenas) con la otra cifra de un sólo dígito y completando la cifra después (1.243+18; 43+18=61; por tanto, 1.261); si las cifras son de dos, o a lo sumo tres, dígitos hará la cuenta mentalmente pero reteniendo en la memoria el «arrastre» necesario (es decir, 145+378 hay quién lo calcula globalmente como 523, pero lo usual es razonar «ocho y cinco son trece —tres y llevo una—, cuatro y siete, con una que llevo, hacen doce —dos y llevo una— que con otra del 145 y tres

más dan cinco: quinientas veintitrés»); por último, si es una expresión larga realizará el cáculo con papel y lápiz escribiendo un número debajo del otro y sumando como lo hace el ordenador. Se podria construir un «mini sistema experto» para realizar este proceso:

R1 SI una cualquiera de las cifras es de un solo dígito
O ambas cifras son de un solo dígito,
ENTONCES calcular la suma de ambas.

SI alguna de las cifras es de mas de dos dígitos
Y la otra de un solo dígito
ENTONCES separar unidades y decenas
Y sumar con la otra cifra
Y completar el número final

SI ambas cifras tienen mas de un dígito
Y ambas cifras tienen menos de cuatro dígitos
ENTONCES sumar las unidades, etc...

Naturalmente, este «mini sistema experto» no tiene sentido porque es bastante más complicado programarlo que dar al ordenador, sencillamente, las dos cifras a sumar (para lo que el ordenador está perfectamente preparado) y porque el proceso de cálculo es perfectamente conocido (en cierto modo, cuando sumamos cifras largas trabajamos al estilo del ordenador) y por muchas razones más, (aparte el hecho de que el sistema tal como se ha propuesto no sería válido: habría que modificar su sintaxis para poder ser procesado). Sin embargo, entendemos que es interesante porque muestra el tipo de razonamiento que se suele hacer en la preparación de un sistema experto: la característica fundamental que queremos subrayar es que en el ejemplo propuesto no nos hemos ocupado de cómo puede hacer el proceso el ordenador ni del «formato» que tenían que tener los sumandos, ni dónde se iba a almacenar el resultado, ni ... de todas esas cosas que hay que ocuparse en la programación tradicional de un ordenador electrónico; sencillamente, hemos reflexionado sobre cómo hacemos nosotros esas tareas y se lo hemos «explicado» al sistema con un lenguaje concreto (reglas de producción) pero bastante inteligible (todas las reglas anteriores se adecuan al esquema: SI «se cumple esta condición» ENTONCES «realiza esta acción»); a partir de esta información, el sistema saca sus conclusiones.

PROGRAMACION CONVENCIONAL VERSUS INGENIERIA DEL CONOCIMIENTO

La Ingeniería del conocimiento es la rama de la Inteligencia Artificial que se ocupa precisamente de los modos de representación del conocimiento (los datos, las informaciones) y su adecuado procesamiento. Los resultados prácticos del trabajo de los «ingenieros del conocimiento» son los «sistemas basados en el conocimiento» o «sistemas expertos». Profundizaremos en estos aspectos al comienzo del próximo capítulo, pero es útil comentar ahora las diferencias y similitudes de estas diferentes técnicas de programar.

Normalmente, como ya hemos comentado, está indicada la creación de un sistema experto cuando el proceso no está bién definido o la información a manejar en cada toma de decisión es muy variada (que no es el caso de la suma, que hemos tomado como ejemplo).

En estas condiciones, se incluyen en el sistema no sólo las reglas de producción propiamente dichas, sino reglas que aportan «estrategias» de búsqueda o reglas de elección (lo que se llaman «heurísticas»): estas reglas incluyen información del tipo «SI el paciente tiene fiebre alta ENTONCES empezar examinando las reglas que consideran las características de los cultivos de laboratorio», que en la parte de la acción indica qué reglas han de ser utilizadas en primer lugar para el establecimiento del diagnóstico.

Es importante observar, además, que estas «metareglas» («reglas de manejo de reglas») hacen referencia a las reglas primarias de la base de conocimientos por su contenido (en el caso anterior, se indica aplicar ciertas reglas del diagnóstico que aluden a unas pruebas concretas) en vez de aludir a ellas por su nombre, como se hace en la programación convencional (alusión a las «etiquetas» de las instrucciones o nombres de los procedimientos).

Esto hace que no haya que reprogramar si se añade una nueva regla y, por tanto, que el sistema pueda ir incrementando automáticamente su propio acervo de conocimientos: el sistema «aprende».

De un modo un tanto simplista, se puede decir que si el sistema incluye una gran base de datos y un «pequeño» algoritmo de tratamiento de datos (los clásicos procesos repetitivos) el esquema óptimo es el de un proceso convencional de tipo «procedural»; si por el contrario, la base de hechos a considerar es pequeña (en un sistema de diagnóstico, sólo los síntomas) pero la información sobre el proceso es amplia y no bien estructurada (una gran base de reglas alternativas o de aplicación parcial y condicionada) el esquema a desarrollar es el de un sistema experto. Una de las ventajas básicas que aporta un sistema de este tipo es la flexibilidad y, de hecho, el nivel de eficacia del sistema («performance») viene dado más

por la bondad de las metareglas que se incluyen (para la búsqueda inteligente de estrategias de proceso) que por las propias reglas de proceso que incorpore.

Por otro lado, es significativo constatar que los programas convencionales son «mantenidos» por programadores (bien que en ocasiones con asesoramiento de los expertos externos) mientras que los sistemas basados en el conocimiento son mantenidos directamente por los expertos, con la colaboración puntual del ingeniero del conocimiento. Esto se debe, entre otras cosas, al hecho de que la propia base de conocimientos de un sistema experto es directamente legible por un no-informático (son reglas del tipo de las anteriormente mencionadas que, en ocasiones, ha escrito directamente el experto —no el ingeniero del conocimiento—); además la propia estructura del sistema hace que sean fácilmente modificables y ampliables por el experto.

Otra característica distintiva (ya apuntada) de los sistemas basados en el conocimiento es que la estructura básica de ellos viene dada fundamentalmente por las reglas de tipo heurístico que incorporan, mientras que un programa convencional se vincula a un algoritmo y de él obtiene la estructura completa del proceso a realizar.

Además de las diferencias de estructura de un sistema experto respecto a un proceso algorítmico convencional, hay que añadir que para la programación del sistema experto se utiliza de base la programación mediante un lenguaje simbólico (LISP, PROLOG, etc.): este hecho comporta algunas diferencias adicionales respecto de cualquier proceso definido mediante programación convencional en un lenguaje procedural.

Entre otras, se pueden señalar las siguientes características distintivas:

— En la programación convencional se manejan datos estructurados en una Base de Datos accesible por direcciones numéricas, mientras que en la programación simbólica se utiliza una base de conocimientos estructurada simbólicamente en un espacio de memoria global.

— La programación convencional está orientada hacia el proceso numérico, en contraposición a la programación simbólica orientada al proceso simbólico (manejo de cadenas de caracteres y procesamiento de sus elementos).

— El proceso suele ser secuencial, por lotes y la secuencia de proceso fija (excepto saltos predefinidos) mientras que en la programación simbólica se obtiene un proceso altamente interactivo y flexible.

— En la programación convencional no es usualmente posible la «explanación durante la ejecución» en contraposición a la facilidad que existe en la programación simbólica para realizar esta explanación mientras se produce el proceso.

RELACION CON OTRAS AREAS DE LA INFORMATICA

Naturalmente, los sistemas basados en el conocimiento o sistemas expertos no son una «isla» dentro del complejo mundo de conocimientos, técnicas y teorías que conforman la Informática. Se suele entender un sistema experto como un sistema que maneja informaciones y nos facilita (mediante unos procesos adecuados) la toma de decisiones o es capaz de elaborar informaciones a partir de las ya obtenidas: es decir, un sistema de manejo de una base de conocimientos. Sin embargo, se puede utilizar programación simbólica para el procesamiento de informaciones de muy diferentes características: por ejemplo, para el manejo, análisis, tratamiento de elementos gráficos.

Así, los sistemas de visión computarizada deben tomar decisiones sobre elementos bastante inconcretos y pueden necesitar estructuras de diálogo con el usuario semejantes a las que se utilizan en los sistemas expertos. El procesamiento de la palabra debe resolver problemas de interpretación de elementos en su contexto mediante técnicas equivalentes a las utilizadas en los sistemas expertos de toma de decisión: es usual, incluso, que el propio sistema experto disponga de un módulo de interface con el usuario para establecer un diálogo cómodo y flexible en la adquisición del conocimiento.

Por otro lado, en ocasiones se incluyen módulos de toma de decisiones o de manejo de datos flexibles en otros sistemas y estos módulos constituyen un pequeño sistema experto en el área específica de que se trate: se puede incluir un módulo de este tipo, por ejemplo, en un sistema de monitorización de una planta industrial y realizar con él diversas operaciones muy útiles (desde evaluar situaciones y recomendar acciones a tomar, hasta realizar simulaciones en vacio de situaciones reales para entrenamiento del personal que ha de operar el sistema).

Se han desarrollado, también, estructuras de manejo de datos «inteligentes» bien como sistemas autónomos bien como núcleos de operación de sistemas generales de manejo de bases de datos (SGBD): en este sentido se habla de Sistemas de Bases de Datos Expertas. Son sistemas que realizan la selección de los datos no mediante estructuras rígidas de búsqueda sino eligiendo sus caminos según reglas heurísticas definidas como una base de conocimientos normal de un sistema experto.

Por tanto, se puede pensar que los esquemas básicos de representación del Conocimiento descritos en el próximo capítulo y las estructuras de procesamiento de estas informaciones que constituyen las técnicas básicas de construcción del motor de inferencia de cualquier sistema experto son ele-

mentos teóricos válidos en numerosas actividades de la Inteligencia Artificial y no se circunscriben al área concreta de los sistemas expertos.

Nosotros aquí, según el esquema propuesto, nos centraremos en estos sistemas informáticos concretos que se han dado en llamar sistemas basados en el conocimiento o sistemas expertos, como conjuntos casi independientes y con su utilidad concretada en el procesamiento de una base de conocimientos dada.



UTILIDAD DE LOS SISTEMAS EXPERTOS

De todo lo expuesto ya habrá deducido el lector varias conclusiones que nos parece, sin embargo, interesante subrayar:

— los sistemas expertos (y en general las ténicas de procesamiento del conocimiento) suponen un punto de vista diferente, dentro del conjunto de la Informática, respecto de los esquemas clásicos de procesamiento de datos;

— este tipo de enfoque aporta una mayor eficacia de proceso, al menos en el tipo de problemas en que es adecuado utilizarlo (que ya hemos esbozado y concretaremos mas adelante);

— sin embargo, no es la «panacea universal» de la informática. Existen numerosas ocasiones en que estas técnicas no aportan ventaja alguna y otros casos, incluso, en que sería más complicado e ineficaz intentar instalar un sistema experto en vez de un buen programa convencional;

— la estrategia de instalación de un sistema experto es distinta, en gran medida, a la de puesta en marcha de una aplicación informática convencional: la diferencia fundamental es la mayor participación del experto no informático en la definición del sistema y creación de la base de conocimientos;

— el mantenimiento posterior se realiza de un modo más flexible que en una aplicación convencional y, normalmente, por parte de propio usuario (en contraposición a lo que sucede en un sistema informático clásico que lo realizan los informáticos —analistas y programadores—);

— casi siempre, un sistema basado en el conocimiento se concibe como un sistema abierto sometido a constantes actualizaciones y «refinamientos» (o, al menos, se suele mantener así durante un período de tiempo grande, en contra de lo que sucede con las aplicaciones usuales informáticas que sólo se entregan al usuario cuando se han depurado completamente).

AREAS DE APLICACION DE LOS SISTEMAS EXPERTOS

Resumiendo todo lo anteriormente expuesto, se puede decir que las características básicas de un sistema experto son:

— en cuanto a la base de conocimientos, que contenga gran cantidad de conocimientos (así como conocimientos sobre cómo utilizar los conocimientos —metaconocimientos—) que estos conocimientosa sean complejos y de utilización múltiple (según los casos), que estén expuestos de forma modular y declarativa y que sean evolutivos (no formando un conjunto fijado y cerrado);

— en cuanto a la obtención de la solución, que se realice paso a paso (en etapas separadas), que estas etapas puedan ser explicadas, que la estrategia de resolución no sea estática sino dinámica (según los detalles del problema y su contexto) y que el usuario pueda intervenir en el proceso de obtención de la solución guiando al sistema según estrategias y heurísticas propias.

En consecuencia, se puede concluir siguiendo a Michel Gondran (ver Bibliografía) que los sistemas expertos están especialmente adaptados para la resolución de problemas en los que se den las siguientes características:

— se dispone de una gran cantidad de conocimientos acerca del problema a resolver;

— este conjunto de conocimientos disponible no está fijado, sino que está en evolución;

— en ocasiones estos conocimientos son incompletos o dudosos (el experto sólo se atreve a afirmar que eso sea cierto con un determinado factor de certeza);

— estos conocimientos son básicamente de tipo heurístico (más que algorítmico);

— el tratamiento de las informaciones es más de tipo simbólico que numérico;

— se conoce el problema más por informaciones de tipo cualitativo que de tipo cuantitativo;

— es tan importante el procedimiento de obtener la solución como la solución misma;

— se necesita saber cómo se ha obtenido la solución (a veces con detalles muy concretos), además de obtener la solución.

Estas características se dan en numerosas áreas del conocimiento y de la actividad humana. Sin embargo, es más importante en muchas ocasiones el aspecto desde el que se aborda un problema que la tipología del pro-

pio problema: es decir, si a partir de las fichas médicas, por ejemplo, de una serie de pacientes de una enfermedad se pretende establecer un análisis estadístico para conocer la frecuencia de aparición de los diferentes síntomas preestablecidos o el número de veces que se observa un dato que se intuye coadyuvante al desarrollo de la enfermedad, un enfoque convencional informático de mecanización de esas fichas será el óptimo; si, por el contrario, se quiere formular de forma explícita un proceso de diagnosis de la enfermedad a partir de ciertos síntomas, evaluando la validez de esta hipótesis y su adecuación a la realidad «objetiva» conocida, la aproximación a través de la constitución de un sistema experto parece más adecuada.

A pesar de ello, se pueden indicar ciertas áreas en las que parece que está especialmente indicada la aplicación de sistemas expertos (y en la mayoría de los cuales, de hecho, se ha desarrollado algún prototipo o sistema real que esté en funcionamiento):

— Detección de averías: en todas las ramas de la ingeniería (especialmente electrónica, comunicaciones e informática), maquinería minera, instrumentación sofisticada militar y de transporte, etc.

— Análisis de datos: en el área de prospección geológica (por ejemplo, el famoso PROSPECTOR), en servicios financieros, profesionales, en aplicaciones militares.

— Monitorización de procesos en tiempo real: son sistemas muy complejos que se han utilizado especialmente en aplicaciones militares.

— Toma de decisiones: en el área informática, de servicios financieros, en sistemas estratégicos, etc.

— Diseño inteligente. Diseño y fabricación asistidos por ordenador con ayudas inteligentes (especialmente en electrónica e informática).

Según Ishizuka las áreas más prometedoras en cuanto al desarrollo futuro de sistemas expertos son:

— Diagnosis y consulta médica. Existen numerosos sistemas en este área, a los que se alude más adelante; se han desarrollado numerosos sistemas en las técnicas más recientes (sistema de consulta en el área de deficiencias cardiovasculares, sistemas para la selección de antibióticos, de afecciones reumáticas, etc.).

— Control y supervisión de plantas industriales.

— Sistemas gerenciales: de estimación de costes, de gestión de almacenes, etc. Incluso se ha construido un prototipo de «secretaria inteligente» que gestiona los horarios del personal de una Organización.

— Diseño asistido por ordenador.

— Proceso de imágenes.

— Acceso y gestión de bases de datos.

ESTRUCTURA DE UN SISTEMA EXPERTO 3



COMPONENTES DEL SISTEMA

VAMOS a ver concretamente en qué consiste un Sistema Experto (SE) y su estructura interna.

El profesor Edward Feigenbaum (uno de los pioneros de la investigación en I.A. en general y en sistemas expertos en particular y coautor del famoso «Handbook of Artificial Intelligence») define un sistema experto como:

«... un programa inteligente que utiliza conocimiento y procedimientos de inferencia para resolver problemas que son suficientemente complejos como para requerir una importante aportación de experiencia humana para su resolución. El conocimiento necesario para operar a este nivel y los procedimientos de inferencia utilizados pueden ser considerados un modelo de la experiencia de los mejores expertos en el área correspondiente.

El conocimiento de un sistema experto está formado de hechos y heurísticas. Los «hechos» constituyen un núcleo de información que es ampliamente compartida por los expertos en la materia, que está disponible públicamente y sobre la que existe un acuerdo generalizado por parte de estos expertos. Las «heurísticas» son reglas mas personales, más «discutibles», de buen criterio, que caracterizan la toma de decisiones por parte de los expertos en la materia (son reglas de razonamiento plausible, reglas que se podrian llamar de «juicios sensatos»). El nivel de eficacia de un sistema experto es función, sobre todo, del tamaño y capacidad de la base de conocimientos que posee.»

Como se vé, en todo momento se refiere al «conocimiento» como base de los sistemas expertos: en efecto, es usual referirse a este tipo de aplicaciones como «sistemas basados en el conocimiento» y, aparte pequeños matices que los técnicos puedan marcar, son términos de hecho equivalentes para la mayoría de las personas que trabajan en este área.

Los profesionales especializados en el diseño y construcción de siste-

mas expertos se llaman «ingenieros del conocimiento» y la técnica en general, «ingenieria del conocimiento».

El conocimiento que incluye un sistema experto, sin embargo, no es uniforme, sino que está compuesto de varios elementos diversos: en el texto citado se alude a «hechos» y «heurísticas» o reglas de conocimiento. Veremos en seguida esto con más detalle.

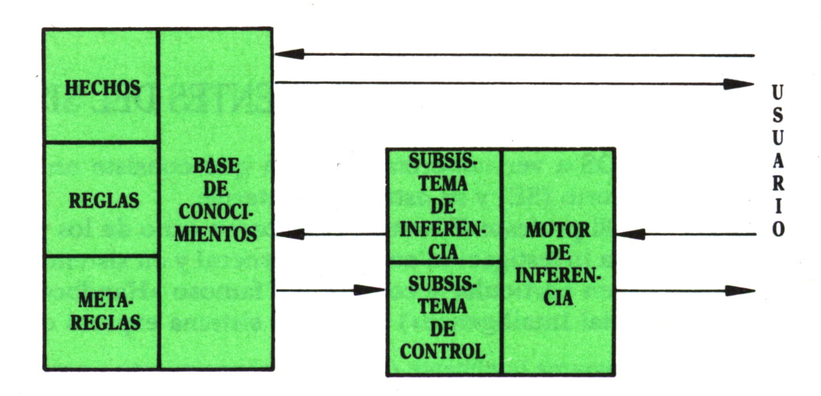


Fig. 2. Elementos básicos de un sistema experto.

Otro grupo de componentes a que hace referencia Feigenbaum, son los procedimientos de deducción. En efecto, este conjunto de procedimientos constituye el segundo elemento básico componente de un sistema experto: el motor de inferencia. La estructura y composición del motor de inferencia puede variar como veremos, pero siempre ha de estar presente en el conjunto, si queremos constituir un Sistema Experto. Por tanto, la estructura básica, mínima, de un sistema experto estará formada por la base de conocimientos y el motor de inferencia. La base de conocimientos tendrá informaciones de diferentes tipos y, dependiendo del sistema de representación del conocimiento utilizado, puede tener estructuras muy diversas. Normalmente incluye tres tipos de informaciones: hechos, relaciones entre hechos y normas de procedimiento. En una base de conocimiento acerca de los animales salvajes se encontrarían hechos como el siguiente: «el animal que estudiamos tiene patas y es de color verde»; también relaciones o reglas del tipo «si un animal es verde, tiene patas y mide mas de dos metros de longitud, es un cocodrilo»; por fin, habría normas de procedimiento o metareglas tales como «si el animal es un carnívoro y su habitat es Africa, estudiar antes que nada los félidos».

El motor de inferencia es un programa que procesa las informaciones contenidas en la base de conocimiento. Normalmente es un mecanismo de estructura bastante sencilla. Su misión es relacionar las reglas entre si y con los hechos para obtener nuevos hechos e ir produciendo deducciones. La propia estructura del motor de inferencia, las meta-reglas incluidas en la base de conocimiento y, en ocasiones, las instrucciones que aporta el usuario guián el «funcionamiento» del motor de inferencia. Ya se ve, por tanto, que hay que examinar dos aspectos al estudiar un motor de inferencia de un sistema experto: su procedimiento de inferencia y el modo en que se produce el control en él.

Normalmente en un sistema experto real intervienen un conjunto de elementos y módulos de proceso adicionales a los ya indicados.

Por un lado es usual incluir algún módulo de adquisición del conocimiento. Este subsistema es útil tanto en la puesta en marcha del sistema experto como en su actualización posterior. Además, el propio funcionamiento del sistema experto suele producir conocimiento adicional. Usualmente, el sistema está diseñado de tal forma que durante el proceso de deducción puede consultar al experto (o al usuario, en la fase de explotación) informaciones adicionales de las que no dispone pero que necesita para continuar su proceso de inferencias.

Por otro lado, una de las grandes ventajas que aporta normalmente un sistema experto frente a otro de programación convencional es su capacidad de explicación de los procesos deductivos efectuados. Para disponer de esta capacidad, hay que dotar al sistema de un módulo adicional que vaya almacenando los datos correspondientes a los procesos inferenciales y sea capaz de reconstruir el proceso deductivo a petición del usuario o del ingeniero del conocimíento. Lógicamente, para que sea minimamente válido y útil este subsistema debe ser suficientemente flexible como para poder producir sus explicaciones a varios niveles (global o de detalle) y en diálogo con el usuario.

Además, suele dotarse al conjunto de algunas facilidades adicionales de comunicación con el exterior: constituyen el subsistema de «interface» con el usuario.

La arquitectura concreta de cada sistema puede diferir del esquema general expuesto, pero los conceptos básicos subyacentes se adecúan usualmente a los presentados aquí: en ocasiones, algunas de las funciones descritas (explicación de resultados, por ejemplo) se incluyen con el propio motor de inferencia o se combinan con otras (del subsistema de adquisición del conocimiento, quizá) para formar parte del módulo de interface con el usuario, etc.

La base de conocimientos, a su vez, suele tener una estructura más flexible e incluir otras informaciones adicionales en áreas auxiliares de almacenamiento: puede haber una memoria que modelice la situación actual,

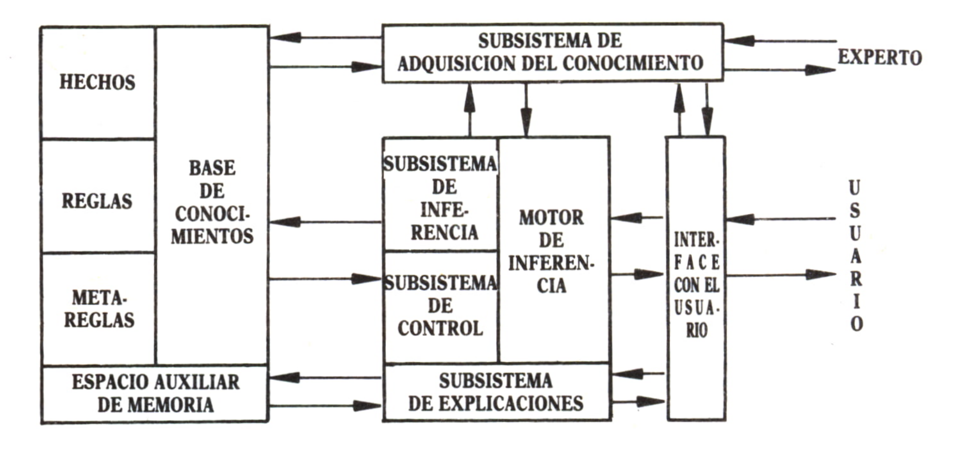


Fig. 3. Arquitectura de un sistema experto.

pueden estar en un área separada los nuevos hechos que va obteniendo el sistema a partir de los que tenía originalmente, suelen guardarse las «trazas» de los procesos deductivos (como ya hemos comentado) para poder construir posteriormente una explicación, etc.

Aunque todos estos elementos descritos suelen aparecer en un sistema experto, las técnicas mediante las que se desarrollan son muy semejantes a las utilidades en cualquier otro sistema informático: la clave de un sistema experto está formada por su base de conocimiento y su motor de inferencia. Por ello, en lo que sigue estudiaremos el modo de construir una base de conocimientos (es decir, los sistemas usuales de representación del conocimiento) y la estructura de un motor de inferencia (o sea, las técnicas de inferencia y de control utilizadas en los motores usuales).



REPRESENTACION DEL CONOCIMIENTO

Al hablar del conocimiento incluido en una base de conocimiento de un sistema experto es necesario distinguir entre las informaciones factuales (acerca de «hechos») y las reglas deductivas (aportan información acerca de como obtener nuevos datos o «hechos» a partir de los que se dispone).

Por otro lado, los diferentes procedimientos disponibles para representar el conocimiento suelen ir vinculados a diferentes lenguajes o sistemas

de representación, con lo que es necesario conocer las cualidades implícitas al sistema elegido para hacer un eficaz uso de ellas.

Uno de los procedimientos básicos de representar el conocimiento es el de las REDES SEMANTICAS, de donde derivan las «TERNAS objeto-atributo-valor» y las «PAREJAS atributo-valor» así como (en cierto modo) los MARCOS o «frames» y GUIONES o «scripts»; otro de los procedimientos básicos es el proporcionado por las EXPRESIONES LOGICAS (cálculo de proposiciones y lógica de primer orden). Veamos en qué consisten cada uno de estos procedimientos.



Redes semánticas

Es un esquema genérico implementado en numerosos lenguajes y herramientas. Originalmente fue propuesto para el análisis semántico, pero la simplicidad relativa con que se pueden obtener deducciones correctas una vez ha sido generada la red es, sin duda, una de las razones básicas que justifican su enorme desarrollo actual (aunque, históricamente, es uno de los primeros esquemas de representación utilizados). La red está constituida por un conjunto de objetos o elementos llamados «nodos» conectados mediante unos «arcos» o «ligaduras». Tanto los arcos como los nodos son elementos básicos de la red y están etiquetados.

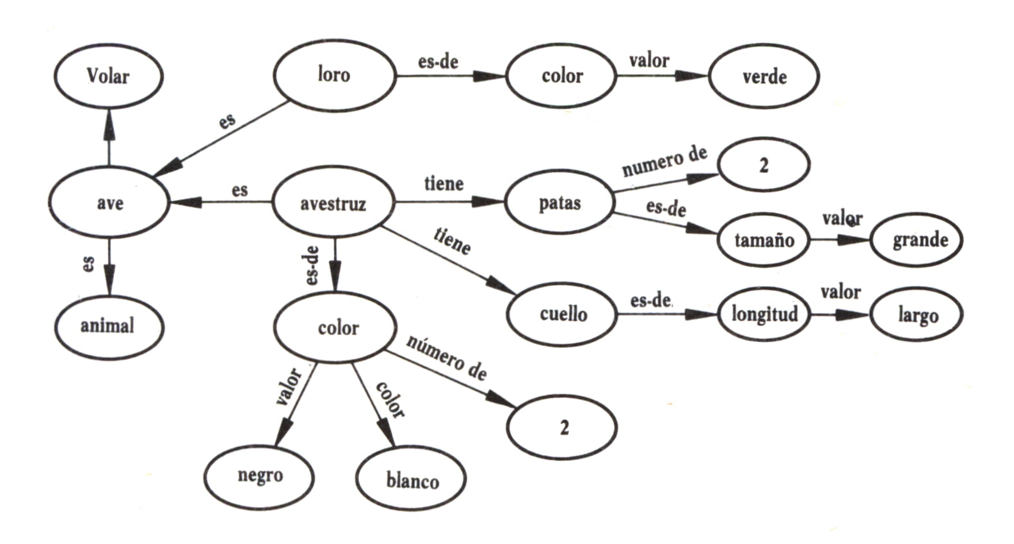


Fig. 4. Una red semántica.

En principio no existen restricciones en el uso de las redes semánticas: los nodos pueden ser tanto objetos físicos (el avestruz, el cuello, las patas, etc.), como categorías abstractas (ave, animal, ...), cualidades definibles de los anteriores objetos o categorías (tamaño, longitud, ...), valores que toman las anteriores cualidades (grande el tamaño, verde el color, ...) e incluso descriptores de algún elemento (2 como número de patas, volador para el ave, etcc.). A su vez los arcos también pueden representar ligazones muy diversas entre los nodos. Como se ve en la red semántica propuesta, es usual que aparezca la relacción «es»: esta ligadura representa la relación de «instanciación» o «particularización» de un elemento de una clase. Mediante la ligazón «es» se suelen vincular los elementos a la «clase» de los elementos que los engloba; así, aparece que «avestruz es ave» igual que «loro es ave» y del mismo modo ave es una instanciación de una superclase que engloba a las aves y otros animales («ave es animal»).

También es usual que aparezca la relación «tiene». Los elementos vinculados mediante la relación «tiene» son partes del elemento que aparece en el nodo ligado. Así, el avestruz tiene patas, tiene cuello, etc. Pero esta relación es distinta de la instanciación: las patas no son un caso, un ejemplo del avestruz; del mismo modo, un ave no está compuesta de avestruces, loros, etc.

Aparece numerosas veces también la relación «es-de» que liga un objeto de un nodo con los nodos donde aparecen las «cualidades» que el objeto tiene: por ejemplo, «patas» es-de «tamaño» (grande), «cuello» es-de «longitud» (larga), etc.

Otra relación genérica es la relación «valor» que liga una cualidad con los posibles valores que adopta (así, el tamaño de las patas es grande, el color del loro verde, etc).

Aparecen, por último, otro conjunto de ligazones diversas que representan otras propiedades o informaciones acerca de los elementos de la red: el ave «puede» volar, el «número-de» patas es 2, etc.

Una de las mayores ventajas de la representación mediante redes semánticas es la enorme flexibilidad que aporta a la descripción del conocimiento. En efecto, por su propia naturaleza, es admisible ampliar en cualquier momento la red mediante la adición de nuevos nodos con sus arcos de ligadura correspondientes.

Otra de las grandes ventajas es la «herencia» o transmisión de cualidades a lo largo de la red.

Si en la red propuesta, desarrollamos toda la información de que dispongamos sobre los «animales» o sobre las «aves» (como hemos hecho con el nodo «volar»), toda esa información queda disponible para ser aplicable a todas las instancias de la clase «ave» (en nuestro caso «loro», «avestruz», etc.) de tal modo que una vez definida la estrructura básica de la red, la

ampliación de conocimientos sobre una clase produce automáticamente la ampliación de información sobre las instancias de esa clase, sin que tenga que ser repetida la información: se evita así mucha redundancia de conocimientos.

La herencia, por el contrario, tiene el inconveniente de que dificulta el establecimiento de excepciones (por ejemplo, si queremos representar el hecho de que el avestruz no vuela a pesar de ser un ave).

Por otro lado, la definición de la red semántica supone una cierta estructuración del conocimiento, pero que «viene dada», no que es impuesta desde el exterior a partir de un proceso de abstracción.



Ternas objeto-atributo-valor

Una forma especial de red semántica aparece en el esquema de ternas o tripletas «objeto-atributo-valor». El conjunto de conocimientos de que disponemos se organiza en forma de una colección de ternas de elementos. En ellas, el objeto puede ser un elemento físico o una entidad abstracta o conceptual; el atributo representa una cualidad o característica del objeto; el tercer elemento de la terna es el valor que, en un momento dado toma el atributo que aparece en la terna, referido al objeto de que se trata.

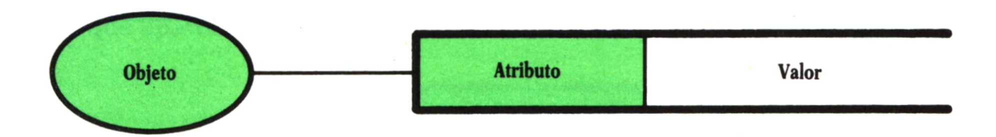


Fig. 5. Terna de valores «Objeto-Atributo-Valor».

Así, a partir de los conocimientos que representabamos en la red semántica descrita anteriormente, se podría establecer la serie de ternas que se incluyen en la figura.

Este modo de representar los conocimientos es menos flexible (más estructurado) pero puede ser más claro. Por otro lado, normalmente será útil incluir alguna información adicional sobre cómo están relacionados

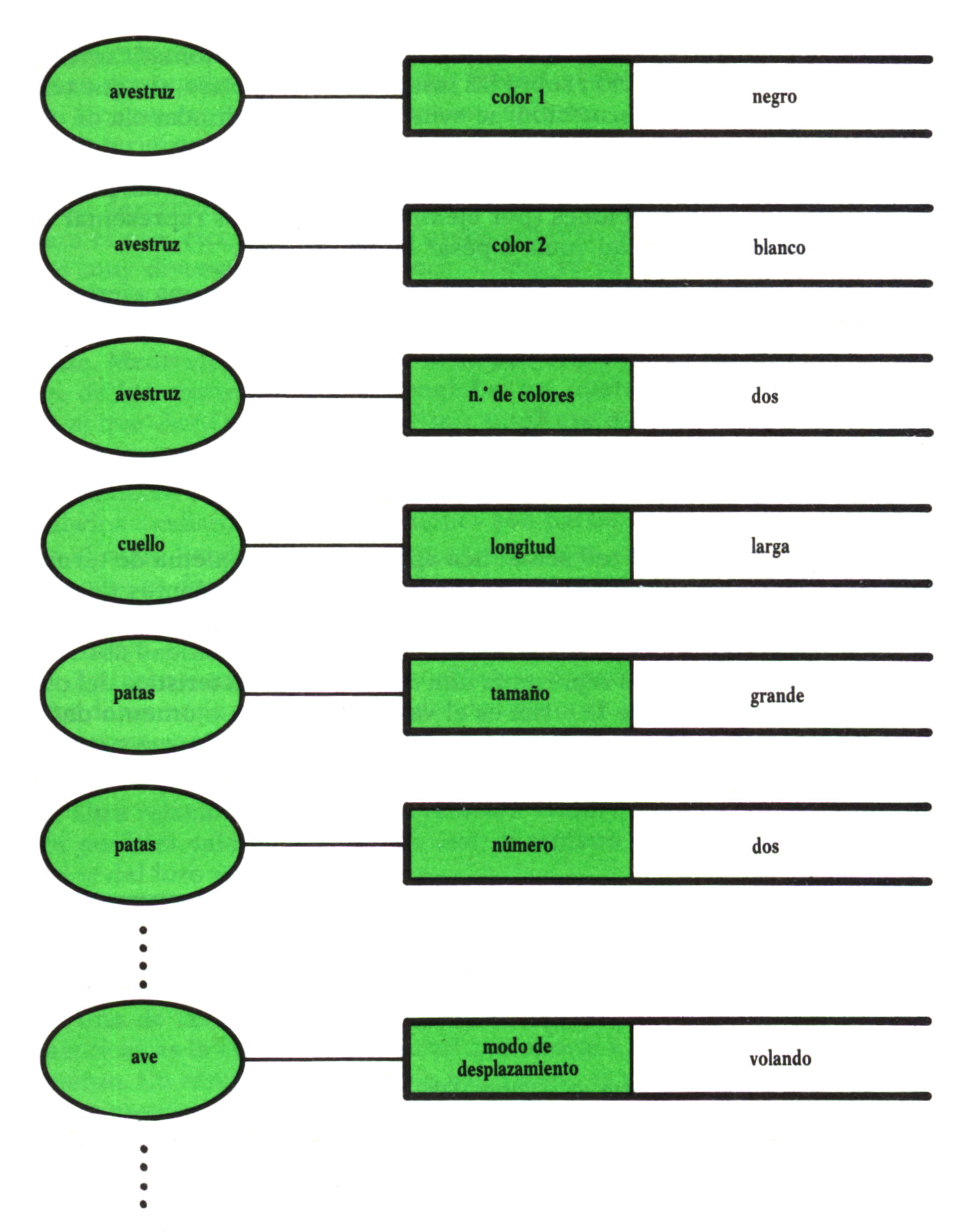


Fig. 6. Conjunto de ternas de la red semántica.

los objetos entre sí. En el caso que estamos tomando de ejemplo, se podría construir un árbol que relacionara los objetos de las ternas, tal como el que se muestra en la figura 7.

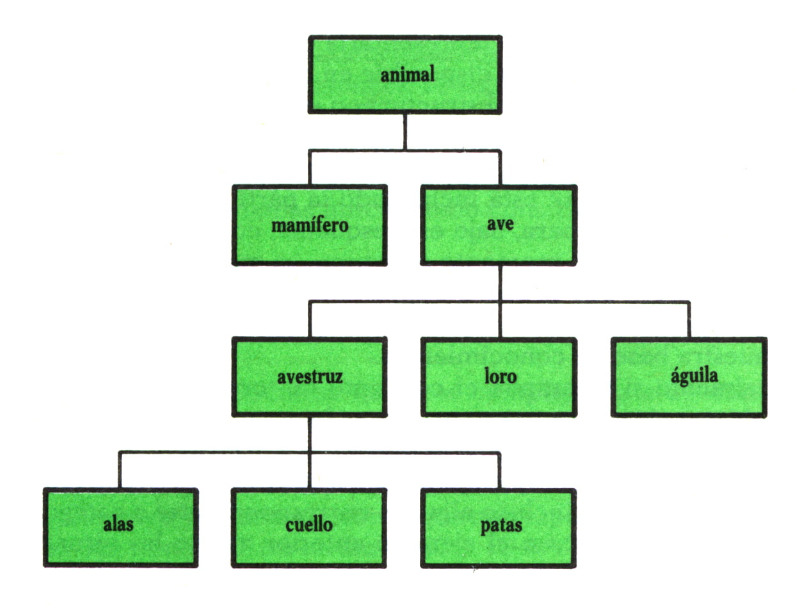


Fig. 7. Arbol estructural de los objetos.

El vínculo (implícito) establecido entre el atributo y el objeto es de tipo «es-de» y el que existe entre el atributo y su valor es, precisamente, de tipo «valor». Las relaciones de instanciación (el ave puede ser un avestruz, un loro, un aguila,...) quedan, en principio, fuera de la tripleta y hay que establecerlas externamente. Lo mismo se puede decir de la relación de pertenencia o «ser-parte-de» (el avestruz tiene «patas», «cuello», etc.).

En ocasiones se puede establecer un esquema genérico de ternas en las que aparecen los objetos y sus atributos pero sin asignación alguna de

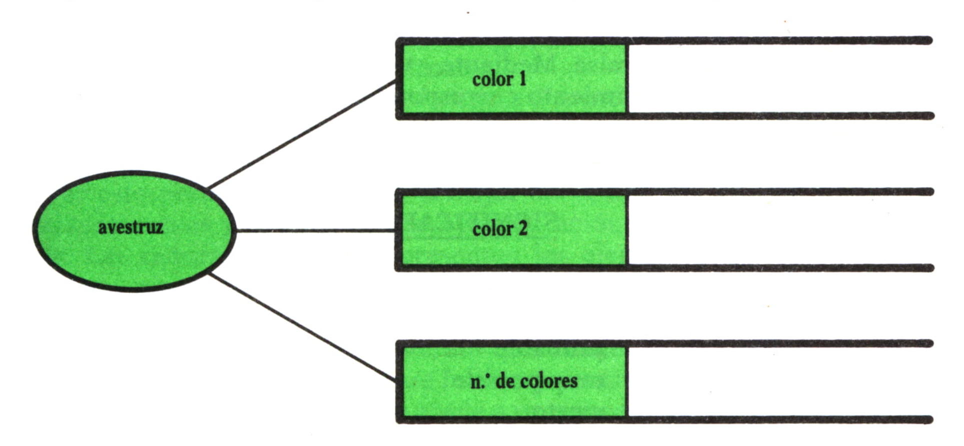


Fig. 8. Infraestructura (o estructura estática) de la base de conocimientos.

valores. Este grupo de ternas representaría el conjunto de informaciones que debe estar (o suele estar) disponible en el sistema, sin referirse a ningún caso concreto. Es la «infraestructura» o «estructura estática» de la base de conocimientos.

Cuando se «llenan» los espacios correspondientes de las ternas (cuando se asignan valores), se está incluyendo la parte «dinámica» de la base de conocimientos. Se opera, bajo este esquema, con la representación estática de la base y, en el momento oportuno, se manejan los valores dinámicos: se establece una primera asignación de valores, pero en el transcurso de la operación se obtienen o facilitan valores adicionales que enriquecen nuestra base de conocimientos.

Naturalmente si se amplia el concepto de terna para representar no sólo los objetos sino categorías superiores la estructura se hace más flexible pero puede ser mas difícil su manejo.

Es usual, sin embargo, que aparezcan ternas con información de objetos a distintos niveles e incluso algunas correspondientes a categorías que engloban las anteriores: en el ejemplo anterior aparte las ternas correspondientes al avestruz o al loro, aparecerían valores para ciertos atributos del objeto «ave».

El tipo y nivel de representación utilizado vendrá dado, en multitud de ocasiones, por el entorno en que estemos moviéndonos y las herramientas de que dispongamos.

El esquema de representación del conocimiento mediante ternas es especialmente idóneo para el manejo de informaciones con incertidumbre (razonamiento impreciso).

Si la asignación de valores a los atributos no se puede hacer de un modo definitivo, se puede afectar cada instanciación de una terna de un coeficiente que indique el grado de certeza de la afirmación correspondiente. Se suele utilizar una escala desde «−1» hasta «+1» para definir estos coeficientes. Con el extremo «−1» representamos la certeza de que la afirmación correspondiente es falsa. Mediante «+1» se suele representar la afirmación tajante del conocimiento correspondiente. Se podrían asignar expresiones del lenguaje natural a cada valor intermedio de la escala y representar las frases del siguiente modo:

VALOR	SIGNIFICADO
0-0'2	desconocido
0'3-0'4	posible
0'6	probable
0'8	muy posible
1	cierto

y las expresiones correspondientes para los valores negativos.

En los casos sencillos, los conocimientos de que dispongamos pueden referirse sólo a un objeto y, entonces, se suprime el elemento «objeto» de la terna, pasando ésta a ser, simplemente, una «pareja» de atributo-valor.

Aunque parece una limitación poco interesante, puede ser útil en ocasiones por la simplicidad que introduce y el aumento de eficacia que aporta. De hecho, existen sistemas funcionando construidos bajo este esquema de representación.



Marcos («Frames»)

Los marcos son estructuras de datos complejas que aportan conocimiento sobre situaciones estereotípicas. Los elementos básicos son los objetos a los que se vinculan informaciones con ellos relacionados. En su forma más elemental el marco representativo de un objeto aporta un conjunto de «casilleros» o «ranuras» («slots») que corresponden a los atributos utilizados en las ternas objeto-atributo-valor (O-A-V). Estas ranuras se «llenan» con los valores correspondientes de los atributos. En principio, por tanto, parece que un marco no aporta demasiado sobre la posibilidad de reunir varias ternas A-O-V correspondientes a un mismo objeto. Sin enbargo, el concepto de marco es más amplio y supone la fijación de una situación estereotípica (y no sólo un cúmulo de datos sobre un objeto). además, se acepta comúnmente que en cada ranura de un marco se pueden establecer unos valores «por defecto», de aplicación inicial en tanto no se obtengan otros valores concretos (el número de colores del ave se puede fijar «por defecto» en 1, aunque en el momento de la instanciación de un marco aparezca un ave con dos colores). También es usual incluir en alguna ranura referencias («pointers») a otras ranuras o a otros marcos, conjuntos de reglas aplicables o incluso procedimientos. Precisamente esta flexibilidad de aportar informaciones y referencias variadas en un marco es lo que les ha hecho muy utilizados.

En el marco presentado en la figura 9 a modo de ejemplo, aparece el nombre de referencia del marco en la parte superior. La primera ranura indica que este marco es una instanciación o especialización del marco AVE: esto permite obtener las ventajas de «herencia» en este tipo de representaciones y establecer una estructuración entre los diferentes marcos. Las ranuras que siguen presentan los elementos que componen el marco y sus posibles valores; en algunas de las ranuras se indica el tipo de información que ha de incluirse, en otras se da la gama de valores entre los que debe estar el valor concreto que «llene» la ranura; en una de las ranuras se incluye una regla que hace referencia a otra ranura del mismo marco; en la última, por fin, se alude a un procedimiento incluido en otro marco.

Como se ve, en un marco está previsto incluir informaciones descrip-

Nombre	Aves 2
Instanciación de	Ave
Nombre	nombre
Tamaño	pequeño, normal grande
Patas, número	valor por defecto: dos
Patas, tamaño	pequeñas, normales grandes, muy grandes
cuello, longitud	número (cm.)
colores, n.°	1 a 4
Colores	blanco, negro verde, amarillo, rojo. Contar y comparar con «colores, n.°»
Hábitat	Averiguar tipo de comida, tamaño y acceder a HABITAT

Fig. 9. Marco vacío (o esqueleto) correspondiente al 2.° tipo de aves.

tivas (representación declarativa de datos) e información de procedimientos (representación procedural). Este fue uno de los aciertos de Minsky cuando en 1975 propuso como esquema de representación los marcos: convinan en un solo esquema la representación de tipo procedural con la representación declarativa.

Otra de las ventajas que han hecho que la representación del conocimiento en marcos se halla extendido tanto es la facilidad de su representación en LISP. En efecto, se puede establecer facilmente una «lista de propiedades» tal que la «propiedad» corresponda a la ranura y el «valor» corresponda al contenido de dicha ranura. Posteriormente, en las sucesivas y diversas implementaciones de este tipo de estructura se ha modificado el esquema básico según las necesidades concretas de cada caso.

Existen varios lenguajes específicos para la manipulación de los marcos: FRL («Frame Representation language», «Lenguaje de Representación de Marcos» -Poberts y Goldstein, 1977) parte de la organización gerárquica de los marcos, lo que comporta relaciones de clasificación y generalización; KRL («Knowledge Representation Languege», «Lenguaje de Representación del conocimiento» -Bobrow y Winograd, 1977) incluye el concepto de «agenda» para el manejo de los procesos y establece cómodos mecanismos de relación de los objetos; UNITS, paquete de diseño utilizado por el sistema MOLGEN (experimentación genética), etc.



Guiones («Scripts»)

Son estructuras genéricas de descripción de situaciones (con articulaciones internas semejantes a las de los «marcos») que representan sucesiones de situaciones tipo. Se han utilizado en el análisis de textos y tratamiento del lenguaje natural. Una «escena» está constituida por campos y contiene acciones que evalúan los valores que satisfacen cada campo. Existe una fuerte interconexión entre los campos debido a la estructura intencionalmente secuencial de la descripción. Se incluyen «planes» que describen en cada escena las opciones posibles para conseguir el objetivo deseado.

La descripción de historias mediante guiones de este tipo, como base previa al análisis de textos, ha sido muy utilizada.



Expresiones lógicas

La utilización de expresiones lógicas supone una aproximación al problema de la representación del conocimiento diferente, en alguna medida al menos, de las anteriormente descritas. Mediante la «lógica de proposicione) se representan los hechos de un modo formalizado en unidades elementales (proposiciones) que sólo pueden adoptar uno de dos valores: verdadero o falso; «el avestruz tiene dos patas», es un ejemplo de proposición simple verdadera.

Las proposiciones simples se combinan entre si mediante un conjunto de «conectivas» muy restringido; se utiliza la conjunción («Y», su símbolo es «$\wedge$»), disyunción («O», símbolo «$\vee$»), la negación («NO», símbolo «$\neg$») y la implicación («IMPLICA», símbolo «$\rightarrow$»). De este modo formalizamos expresiones del tipo «el avestruz tiene dos patas **Y** es de color negro **O** es de color blanco» (que simbólicamente se escribirá $p \wedge (q \vee r)$, siendo $p \equiv$ «el avestruz tiene dos patas»; $q \equiv$ «el avestruz es negra», $r \equiv$ «el avestruz es blanca») o bién, «que el avestruz sea un ave **IMPLICA** que puede volar» ($s \rightarrow t$, con $s \equiv$ «el avestruz es un ave»; $t \equiv$ «el avestruz puede volar»). Como

se ve, en el primer caso se han combinado en una proposición tres afirmaciones (proposiciones simples) que pueden corresponder a tres ranuras de un marco o a tres arcos de una red semántica, afirmando la necesidad (caso del Y) o la opcionalidad (caso del O) de que sean ciertas las proposiciones componentes para que lo sea la proposición compuesta de todas ellas. En el segundo caso, se afirma que el hecho de que el avestruz sea un ave supone que tiene la propiedad de volar como todas las aves; se está afirmando la herencia de las propiedades del ave (una de ellas, en concreto) por parte del avestruz (instancia de la clase «ave»), en vez de establecer la relación jerárquica de pertenencia del avestruz a la clase correspondiente.

En la lógica de proposiciones se manejan las frases del lenguaje (las proposiciones) como unidades completas sin formular (formalizar) su «contenido interno»: si en vez de considerar como una unidad la frase «el avestruz tiene dos patas» queremos analizar su contenido, examinaremos que en la frase aparece una «propiedad» (tiene dos patas) que se puede representar por el símbolo DP (en lógica de primer orden se llama «predicado») y que se afirma que esa propiedad la tiene el avestruz (llamémosle «a»): podemos, por tanto, escribir —como se suele hacer en matemáticas con las funciones— que DP(a) (es decir, que se cumple «DP» para el elemento «a»). Esto permite comparar lo que sucede con otro animal («b», por ejemplo) que también tiene dos patas (será verdad, por tanto, que DP(b)), o formular más precisamente la frase anterior: en efecto, la afirmación «el avestruz tiene dos patas» normalmente (dependerá del contexto) se referirá no a un avestruz en concreto sino a cualquier avestruz (es decir, es equivalente a «todas las avestruces tienen dos patas») por lo que se podría formular como $\forall x$ PD(x) («cualquiera que sea “x” del conjunto de las avestruces, se cumplirá que DP(x)») o, incluso, introduciendo el predicado «ser avestruz» como A(x), formalizar $\forall x\ (A(x) \rightarrow DP(x))$ («para cualquier animal el ser avestruz implica tener dos patas»).

Se pueden formalizar mediante predicados no solo *propiedades* de los elementos (como en el caso de A(x) «x es avestruz» o DP(x) «x tiene dos patas») sino *relaciones* entre varios elementos, como cuando escribimos P(x, y) «x es padre de y» o ENTRE (x, y, z) «x está situado entre y y z».

Naturalmente, cada expresión simple en lógica de primer orden (con predicados) puede ser considerada y tratada como una proposición y, por tanto, unida a otras proposiciones mediante las conectivas vistas anteriormente. Por ejemplo, la frase anterior «el avestruz (todas las avestruces) tiene dos patas Y es de color negro O de color blanco» puede formalizarse como $\forall x\ (A(x) \rightarrow DP(x) \wedge (B(x) \wedge N(x))$, haciendo $B(x) \equiv$ «x es blanca»; $N(x) \equiv$ «x es negra». Como se ve, aparte de los predicados que ya conocíamos, los que acabamos de definir y el «cuantificador universal» ($\forall x$) aparecen en la expresión las «conectivas» $\rightarrow$, $\vee$, $\wedge$ uniendo las distintas proposiciones componentes de la frase total.

La introducción de la lógica de primer orden (este segundo nivel de formalización que estamos comentando) con el manejo de predicados que indican propiedades o relaciones, con los símbolos de cuantificación ∀x ≡ «para todo x» ≡ «cualquiera que sea x» y∃x ≡ «existe x» ≡ «hay, al menos, un x que cumple») y la utilización de variables que pueden tomar valores entre los elementos de un conjunto, se puede realizar una formalización de las frases del lenguaje natural más próxima a la realidad, pero el manejo de estas informaciones es más compleja. Existen numerosos sistemas expertos construidos con bases de conocimientos formuladas en lógica proposicional, pero cada día se van extendiendo más los sistemas capaces de manejar también lógica de primer orden.



Reglas de producción

El formalismo de la lógica que acabamos de comentar permite la definición cómoda y precisa de ciertas propiedades que pueden cumplir los elementos de la base de conocimientos o de algunas relaciones que se pueden dar entre ellos. Por ejemplo, podemos establecer la norma siguiente (un tanto simplista) de análisis zoológico «SI un animal es ave, tiene dos patas y es de color blanco o negro ENTONCES es un avestruz». Esta expresión no es mas que una proposición con una conectiva «implicación», cuatro antecedentes («es ave», «tiene dos patas», «es blanco», «es negro») y una conclusión («es avestruz»).

Esta proposición representa una información de tipo deductivo: permite obtener nuevas informaciones a partir de las ya existentes; es decir, si ya sabemos que el animal objeto de nuestro analisis «es ave y tiene dos patas y es o bien blanco o bien negro», la proposición que comentamos nos indica que podemos concluir que «es un avestruz». Cuando se manejan estos conceptos como tales expresiones lógicas, en vez de la proposición anterior con una implicación se suele formular el mismo contenido de información con un símbolo especial «⇒» que se lee «permite deducir». Se escribe «el animal es ave» y «tiene dos patas» y («es blanco» o «es negro»)⇒«es un avestruz».

Estas expresiones que se adecúan a la estructura formal «SI condición1 ⊗ condición2 ⊗ condición3, etc ENTONCES acción1 ⊗ acción2 ⊗ acción3, etc», donde ⊗ puede ser una conjunción («Y», ∧) o una disyunción («O», ∨), se llaman «reglas de producción», debido al carácter deductivo que comportan, que permite «producir» nuevas informaciones a partir de las ya disponibles. Las «condiciones» que aparecen en la primera parte de la regla se suelen llamar, también «premisas» o «cláusulas SI»; las acciones que aparecen a la derecha del ENTONCES (que se corresponde con el símbolo de implicación, «→») se llaman «conclusiones» o «cláusulas ENTONCES». En el antecedente de la regla, las premisas deben ir unidas por

la conectiva «Y» (aunque alguna de estas premisas sea una condición compuesta de otras varias mediante la disyunción «O»). El consecuente de la regla suele tener, en la mayoría de los casos, una sola conclusión.

Aunque el formalismo de las reglas de producción se corresponde con el general de la lógica, este tipo de expresiones se utilizan no sólo en los sistemas que usan como esquema de representación del conocimiento la lógica, sino también en aquellos que se basan en representaciones de tipo red semántica o inferencial, ternas objeto-atributo-valor, marcos, etc. Así, por ejemplo, en un sistema experto famoso (el sistema MYCIN, diseñado para el diagnóstico de enfermedades infecciosas) que utiliza para la representación del conocimiento el esquema de ternas O-A-V, la mayoría del conocimiento del experto está expresado en forma de «reglas de producción» como la siguiente:

SI	la base del cultivo es sangre y la morfología del organismo es bastón y la reacción gram es gramneg
ENTONCES	hay una probabilidad grande (0'6) de que la identidad del organismo sea Pseudomonas-aeruginosa.

Como se ve, en esta regla aparecen tres premisas y una conclusión y, en efecto, la estructura de los cuatro elementos es la de Objeto-Artículo-Valor (Cultivo-Base-Sangre; Organismo-Morfología-Bastón; Organismo-Reacción gram-gramneg; Organismo-Identidad-Pseudomonas aeruginosa).

En la regla presentada, por otro lado, aparece un elemento que suele incluirse con frecuencia en las reglas de producción (al igual que en otros elementos de la base de conocimientos, como ya hemos visto): los factores de incertidumbre que permiten realizar «razonamiento impreciso». La regla dada no se cumple siempre, indefectiblemente (en opinión del experto —el médico—) sino «con mucha frecuencia»: por ello asigna una probabilidad relativamente grande (0'6 sobre 1) al hecho de que sea cierto que la identidad del organismo sea Pseudomonas-aeruginosa, si se cumplen las tres premisas dadas.

Naturalmente, estas reglas de producción pueden incluir expresiones con variables (utilizando el formalismo de la lógica de primer orden). Por ejemplo, en vez de incluir las reglas:

regla 1: SI	Adolfo es padre de Juan Y Juan es padre de Luis
ENTONCES	Adolfo es abuelo de Luis
regla 2: SI	Ramiro es padre de Jonás Y

Jonás es padre de Ramón
ENTONCES Ramiro es abuelo de Ramón
regla 3: SI José es padre de Pedro Y
Pedro es padre de Paco
ENTONCES José es abuelo de Paco
etc.

Se pondría la regla genérica

regla 4: SI X es padre de Y Y
Y es padre de Z
ENTONCES X es abuelo de Z

para, a partir de la base de hechos

Adolfo es padre de Juan
Jonás es padre de Ramón
José es padre de Pedro, etc.

deducir que

Adolfo es abuelo de Luis
Ramiro es abuelo de Ramón, etc.

«intanciando» o particularizando la regla 4 para los valores posibles que cumplan la condición clave de la regla: que el elemento de la derecha de la primera condición (Y) coincida con el elemento de la izquierda de la segunda condición. (Esta es, precisamente, la tarea a realizar por el motor de inferencia —no solo en este caso elemental sino sobre todo en otros mas sofisticados—, como veremos a continuación).



EL MOTOR DE INFERENCIA

El motor de inferencia de un sistema experto es un conjunto de módulos que realizan las tareas básicas de proceso del conocimiento. Existe todo un cúmulo de actividades que debe realizar el Motor de Inferencia (MI) del sistema: proceso de deducción propiamente dicho, resolución de con-

flictos, gestión de prioridades, control del orden de utilización de las reglas, manejo de las memorias auxiliares (dónde se van manteniendo los nuevos hechos deducidos, el estado del sistema y otras informaciones temporales, etc.), cálculo de los factores de incertidumbre «propagados» a través del sistema, etc.

El núcleo del MI es el módulo de deducción. Dependiendo del esquema utilizado en la representación del conocimiento, la estrategia de deducción puede variar. Sin embargo, la mayoría de los sistemas utilizan reglas de producción y su manejo se realiza mediante algunas (en numerosos casos una sola) reglas deductivas de la lógica. La estrategia básica utilizada es la regla llamada de «modus ponens». En los sistemas que utilizan lógica de primer orden (lógica de predicados) se utilizan, en ocasiones otras «reglas» como «silogismo» o «modus tollens», según veremos más adelante.

La regla de «modus ponens» afirma que sí es cierto (se puede afirmar, consta) que se cumple un hecho «A» y tenemos una regla de producción que asevera que «A→B» («de A se deduce B» o «A implica B» o «SI A ENTONCES B»), seguro que se puede afirmar el hecho «B». Esto permite, a partir de los hechos que se conocen cuando comienza el proceso de deducción, obtener nuevos hechos. Naturalmente, si disponemos de otra regla que afirme «B→C», podemos aplicar de nuevo «modus ponens» para añadir un nuevo hecho demostrado (C) en nuestra base de hechos.

La regla de «modus ponens» es sumamente simple y produce razonamientos sencillos y suficientemente potentes (en multitud de casos). Sin embargo, hay procesos deductivos, claros e intuitivos que no pueden hacerse (al menos de un modo directo) con esta regla.

Observemos, por ejemplo, el razonamiento: «Si el animal considerado es un avestruz, entonces es un ave» (ESAVESTRUZ→ES-AVE); «el animal que estamos considerando no es un ave» (NO ES-AVE); por tanto «dicho animal no es un avestruz» (NO ES-AVESTRUZ). Parece bastante claro que es correcto (y es muy simple); sin embargo, no puede ser resuelto directamente por «modus ponens»; habría que «deducir» por algún procedimiento (existe un teorema en lógica que muestra que este paso es correcto) la expresión «si no es ave no puede ser avestruz» a partir de frase dada (es decir: obtener NO ES-AVE→NO ES-AVESTRUZ a partir de ES-AVESTRUZ →ES-AVE) y aplicar después «modus ponens» con «NO ES-AVE→NO ES-AVESTRUZ» y «NO ES-AVE» para obtener «NO ES-AVESTRUZ».

Existen otras transformaciones básicas que estudia la LOGICA y son de aplicación en el procesamiento de expresiones de este tipo, como la equivalencia de «ES-AVESTRUZ→ES-AVE» con «O NO ES-AVESTRUZ O ES-AVE» y con «no puede ser que se cumpla simultaneamente que ES-AVESTRUZ y NO ES-AVE», pero en la mayoría de los casos se utiliza exclusivamente la regla de «modus ponens» para producir las sucesivas deducciones.

Sin embargo, la aplicación de esta regla deductiva no se puede realizar de un modo anárquico con todas las reglas de producción que existan en la base de conocimientos y con todos los hechos que estén demostrados en la base de hechos y/o memorias auxiliares. Deben definirse de algún modo las estrategias de búsqueda para que el proceso sea mas eficaz.



Control del proceso deductivo

Aunque, como se ha visto, el proceso de obtención de una conclusión (mediante aplicación de la regla general de deducción, «modus ponens», a un hecho conocido) es un paso sumamente simple, el proceso general deductivo es más complejo. Hay dos problemas básicos en el funcionamiento del motor de inferencia:

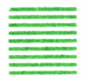
a) ante todo, cómo comenzar el proceso y/o cuál es el objetivo a alcanzar: en efecto, la base de conocimientos es un cúmulo de información estática y habrá que arrancar el proceso de algún modo. Pueden utilizarse procedimientos diferentes: en ocasiones se marca(n) alguno(s) hecho(s) básico(s) con los que el motor debe comenzar su actividad deductiva; en otros casos se marca un objetivo a obtener (un hecho a demostrar) y es tarea del sistema ver si es demostrable o no; hay sistemas que admiten la presentación de un objetivo genérico (un conjunto de varios hechos que pueden ser demostrados) y el sistema elabora un proceso deductivo para ver, a partir de los hechos ya conocidos, cuál es el que se puede demostrar ... ¡veremos esto en detalle a continuación!.

b) por otro lado, el motor de inferencia debe resolver los conflictos que se presentan cuando, en su tarea deductiva, puede elegir varias líneas de razonamiento alternativas (a partir de diferentes reglas que pueden ser aplicables). Debe disponer de «normas» de funcionamiento o «reglas sobre cómo aplicar las reglas»; o bien, debe tener capacidad para consultar al usuario ... o ambas cosas a la vez.

Se puede hablar de unas técnicas generales de conducción del proceso deductivo: encadenamiento adelante o atrás, búsqueda en profundidad o en anchura, utilización de reglas de tipo «ver primero» o heurísticas (control por el contenido de las reglas), etc., aunque existen numerosos procedimientos de control adicionales que se suelen utilizar en unos u otros de los sistemas expertos y/o herramientas disponibles (activación y desactivación de paquetes de reglas o control por «objetivos ficticios» y «datos ficticios» en Intelligence Service; «gestión de hipótesis» y utilización de primitivas de metaconocimiento es SNARK; control de reglas que se activan pasado un umbral de certeza y reglas de uso exclusivamente en encadenamiento adelante en M.I.M.I., etc.).

Veremos en detalle las técnicas generales citadas anteriormente y co-

mentaremos los restantes procedimientos de control para que el lector tenga una visión lo más completa posible de las técnicas utilizadas para la gestión del proceso deductivo en los motores de inferencia.



Encadenamiento adelante y atrás

Si arrancamos el proceso deductivo a partir de un conjunto de reglas y uno o varios hechos conocidos, se puede aplicar la regla de «modus ponens» para ir obteniendo hechos nuevos demostrados. Asi se va «avanzando» a través de la base de conocimientos obteniendo sucesivamente nuevos hechos a utilizar en los pasos posteriores: esto es el encadenamiento hacia adelante.

Si, por el contrario, queremos llegar a una conclusión (objetivo a obtener), se puede examinar qué reglas de producción tienen en su conclusión ese objetivo. Los antecedentes de la o las reglas que cumplen esa condición son los nuevos objetivos (subobjetivos se llaman a veces) a obtener, pues si podemos demostrar estos subobjetivos, automáticamente la aplicación de la regla correspondiente permitirá obtener el objetivo buscado. Con cada uno de los subobjetivos se realiza la misma operación buscando las reglas en que son conclusión: los antecedentes (parte de condición o premisas) de esas reglas darán los nuevos subobjetivos a obtener ... y por este procedimiento vamos «retrocediendo» por la base de conocimientos hasta los primeros hechos que conocemos o sobre los que se pregunta al usuario. Este es el encadenamiento hacia atrás.

Podemos considerar un ejemplo de pequeño sistema experto (*) para ver en detalle estos conceptos:

BASE DE CONOCIMIENTOS

CARACTERISTICAS DE LOS ANIMALES EN GENERAL

Regla 1: si (da leche) → es un (mamífero).
Regla 2: si (tiene pelos) → es un (mamífero).
Regla 3: si (vuela)
y (pone huevos) → es un (ave).
Regla 4: si (tiene plumas) → es un (ave).
Regla 5: si es un (mamífero)
y (tiene pezuñas) → es un (ungulado).

(*) Confeccionado siguiendo el ejemplo de sistema que presentan Winston y Horn en su libro sobre el lenguaje LISP (1981).

DATOS DE LOS PAJAROS

Regla 6: si es un (ave),
que (no vuela),
que (es de color blanco y negro),
que (tiene cuello largo)
y que (tiene grandes patas) → es un (avestruz).

Regla 7: si es un (ave)
y (es de color verde) → es un (loro).

DATOS SOBRE LOS MAMIFEROS

Regla 8: si es un (ungulado),
que (tiene cuello largo),
que (tiene manchas),
y que (tiene grandes patas) → es una (girafa).

Regla 9: si es un (ungulado),
que (tiene rayas),
y que (es de color blanco y negro) → es una (cebra)

Regla 10: si es un (mamífero)
y que (vuela) → es un (murcielago).

Si se dispone, adicionalmente, de la siguiente

BASE DE HECHOS

el animal considerado
vuela
pone huevos
es de color verde

entonces se dispara un proceso de encadenamiento adelante y el motor de inferencia busca en el antecedente de qué regla(s) aparece alguno o varios de los hechos que conocemos. Encuentra que «vuela» aparece en el antecedente de la regla 3. Averigüa entonces qué otros hechos debemos tener como ciertos para poder «disparar» la regla 3. Como el hecho de que pone huevos es, también, otro de los que se encuentra en la base de hechos, «dispara» la regla 3 (aplica el proceso de «modus ponens» con esa regla y los dos hechos considerados) y obtiene la conclusión: «el animal que estamos considerando es un ave». Este nuevo hecho obtenido incrementará nuestra «base de hechos» (será puesto junto a los anteriores en

una única base de hechos o en una memoria auxiliar, dependiendo de los detalles concretos de construcción del sistema experto considerado). A continuación habrá que seguir el proceso de búsqueda con todos los hechos de que disponemos, y se disparará la regla 7, pues disponemos de los hechos «es un ave» y «es de color verde». Por tanto obtendremos que «el animal considerado es un loro». Normalmente, el motor prevé la posibilidad de obtener varias conclusiones (excepto que, por el conocimiento del

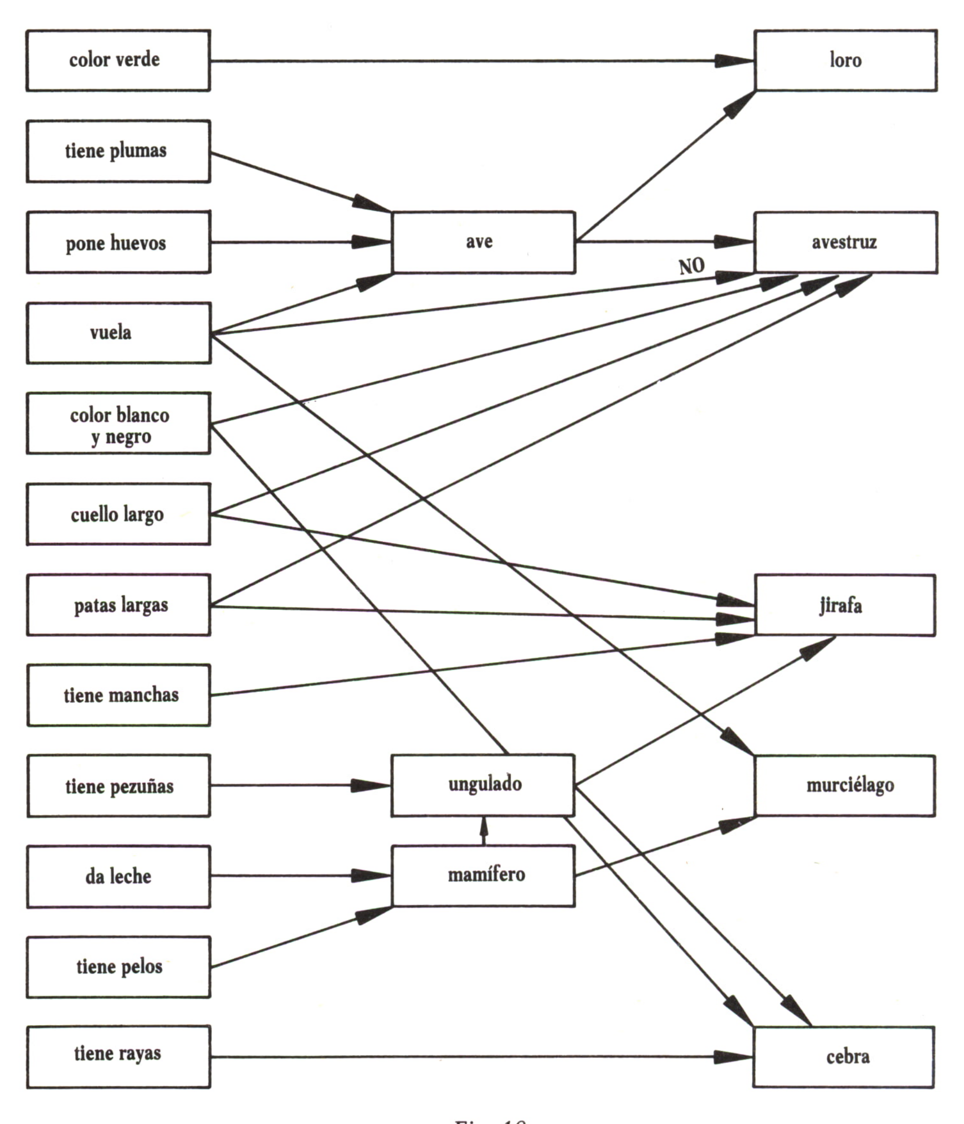


Fig. 10.

problema concreto a resolver, programemos lo contrario) y por tanto, seguirá la búsqueda de nuevas reglas que se puedan aplicar pero, en el ejemplo propuesto, no obtendrá más conclusiones.

Si, por el contrario, nuestro interés es saber si los hechos de que disponemos nos permiten llegar a la conclusión de que «el animal a que se refieren es una girafa», marcaremos al sistema el objetivo «girafa» para que trabaje en «encadenamiento hacia atrás». Dado que girafa aparece como conclusión de la regla 8, el sistema pasa a averiguar si dispone, como ciertos y demostrados, de los hechos «es un ungulado», «tiene cuello largo», «tiene manchas» y «tiene las patas largas». No dispone de estos hechos y, por tanto, ve si puede demostrarlos. Para demostrar que «es un ungulado» debe partir (regla 5) de que «es un mamífero» y «tiene pezuñas» y para ver si es mamífero de que «da leche» (regla 1) y de que «tiene pelos» (regla 2). (Obsérvese que, hasta ahora, todas las condiciones eran de tipo conjuntivo —debían cumplirse todas para poder «disparar» la regla correspondiente—, pero en el último caso, las condiciones para ser mamífero son de tipo disyuntivo —con que se cumpla una de ellas ya se puede aplicar la regla correspondiente: regla 1 o regla 2—).

Por tanto, debemos tener en la base de hechos como ciertos los siguientes datos:

tiene cuello largo tiene manchas tiene las patas largas	para regla 8
tiene pezuñas	para reglas 8 y 5
tiene leche o tiene pelos	para reglas 8,5 y (1 ó 2)

Si están, el sistema concluirá que, en efecto, el animal con esas características «es una girafa».

Si no aparece alguno de los hechos indicados (o ninguno de los dos últimos), normalmente el sistema estará capacitado para preguntar al usuario si se dispone de esos hechos (en la mayoría de los casos se irá haciendo a medida que se va realizando el proceso de encadenamiento hacia atrás: al examinar la regla 8 y ver que no «sabe» si «tiene el cuello largo», lo preguntará; y lo mismo con las manchas y las patas largas). Si alguna de las respuestas a estas tres preguntas es «no», deja de trabajar con la regla 8 y, en este caso concreto, detiene el proceso, puesto que la girafa sólo es conclusión de la regla 8. En nuestro ejemplo, la contestación seria que «no sabe» si el animal es una girafa o que la afirmación «el animal es una girafa» es «desconocida».

Obsérvese que no contesta que es falso, porque como se estudia en lógica, cuando se dispone de una regla de producción «A B», si se cumple A, se puede asegurar que es cierto que B; pero si no se sabe nada de A o se sabe que es falsa, no se puede asegurar nada de B. Normalmente este tipo de sistemas (sistemas de producción en lógica de proposiciones) suelen aceptar tres posibilidades para un hecho: cierto, falso o desconocido.

Se puede dar otro tercer caso: utilizando la base de conocimientos de que disponemos podemos intentar descubrir qué animal es el que hemos visto. No sabemos qué características son las importantes (para aportarlas como hechos), pero podemos contestar a las preguntas del sistema (concretamente, según el animal que hemos visto iremos contestando a las preguntas del sistema, tal como describiremos a continuación: que vuela, no pone huevos, no tiene plumas, da leche y no tiene pezuñas).

Como se ve en la base de conocimientos, los posibles «animales a examinar» (es decir, los posibles objetivos a deducir en el sistema) son cinco (avestruz, loro, girafa, cebra y murciélago). El sistema empieza examinando (en encadenamiento hacia atrás) la posibilidad de que sea un avestruz, salvo que mediante algún otro dispositivo podamos (y queramos) controlar el proceso deductivo, si el sistema concreto está preparado para ello.

Para establecer que es un avestruz (regla 6) necesitaría saber si «es un ave», si «no vuela», si «es de color blanco y negro» ... Como no sabe si es un ave (primer dato que necesita) examinará la base y concluye que necesta saber si «vuela» y «pone huevos» (regla 3) o si «tiene plumas» (regla 4). Como tampoco sabe si «vuela» o no, y ese hecho no aparece como conclusión de ninguna regla debe preguntarlo (si no puede preguntar, acabará la búsqueda con la regla 3 y pasará a la 4). Nuestra respuesta es (según los datos hipotéticos que vamos a utilizar como ejemplo) según hemos comentado que «si vuela». A continuación el sistema pregunta si «pone huevos»: contestamos que «NO pone huevos». Por tanto, no se puede «disparar» la regla 3.

A continuación sigue con la regla 4 y pregunta si «tiene plumas»: nuestra respuesta es NO y, por tanto, no puede concluir que sea un ave (tampoco tiene datos para afirmar que no lo sea) y decide que «no sabe» si es un ave («es un ave» no es ni verdadera=V, ni falsa=F, sino desconocida=D). Por tanto, deduce que no puede saber según la regla 6 si el animal es un avestruz o no («es un avestruz» tiene valor D).

A continuación utilizará los datos que va teniendo para ir deduciendo conclusiones (en encadenamiento hacia adelante). De ese modo, del hecho de que «es ave» sea D debe concluir (regla 7) que no puede saber si es un loro o no («es un loro» toma valor D). Mientras tanto, ha ido tomando nota de que ya ha utilizado (sin éxito, por cierto) la regla 3, la 4, la 6 y la 7: no tiene sentido volver a examinarlas cuando sabemos que tienen en sus premisas hechos desconocidos=D (si incluimos algún hecho nuevo, volverá a comenzar, borrando la memoria de trabajo).

Encadenamiento hacia atrás	Inferface con el usuario		Encadenamiento hacia delante		
Búsqueda de hechos	Respuestas	hechos establecidos	conclusiones	Reglas	
¿Es una avestruz? (regla 6) ↓					
¿Es un ave? ↓					
Pregunta: ¿vuela? (regla 3)	SI	Vuela = V	(ninguna)	3	
Pregunta: ¿pone huevos? (regla 3) ↓	NO	pone huevos = F			
Pregunta: ¿tiene plumas? (regla 4)	NO	tiene plumas = F	es ave = D es avestruz = D es loro = D	4 6 7	
¿es una jirafa? (regla 8) ↓					
¿es ungulado? ↓					
¿es mamífero? (regla 5) ↓					
Pregunta: ¿da leche? (regla 1)	SI	da leche = V	mamífero = V murciélago = V		1 10
Pregunta: ¿tiene pezuñas? (regla 5)	NO	tiene pezuñas = F	es ungulado = D es jirafa = D es cebra = D	5 8 9	
				(La regla 2 queda sin ser utilizada)	

Fig. 11.

Como el proceso de deducción por encadenamiento hacia adelante «no da mas de sí», sigue haciendo la búsqueda hacia atrás con la primera regla no utilizada (la 8) para ver si el animal es una girafa. Para ello, debe saber si «es un ungulado» y para esto (regla 5) si «es un mamífero». Las reglas que se pueden utilzar para saber si un animal es un mamífero son la primera y la segunda: pero el sistema no tiene como hechos demostrados (o conocidos al menos) ninguno de los antecedentes de estas reglas. Por tanto, debe preguntar «¿da leche?»; nuestra respuesta será qie «SI da leche»: por tanto, «es un mamífero» (regla 1).

A continuación el sistema utiliza los nuevos datos de que dispone («da leche» y «es un mamifero») para —mediante encadenamiento hacia adelante— obtener mas conclusiones: así, encuentra que dispone ya de una premisa de la regla 5 (la otra no) y ¡dispone de las dos premisas de la regla 10!. Estas dos premisas son verdaderas, además, por lo que puede concluir que el animal «es un murciélago», aunque no sea ésta la conclusión que trataba de demostrar.

Normalmente, excepto que se programe lo contrario, el sistema seguirá trabajando y preguntará (regla 5) si «tiene pezuñas». Nuestra respuesta es NO y, en consecuencia, no sabemos si «es ungulado» ni, consiguiente-

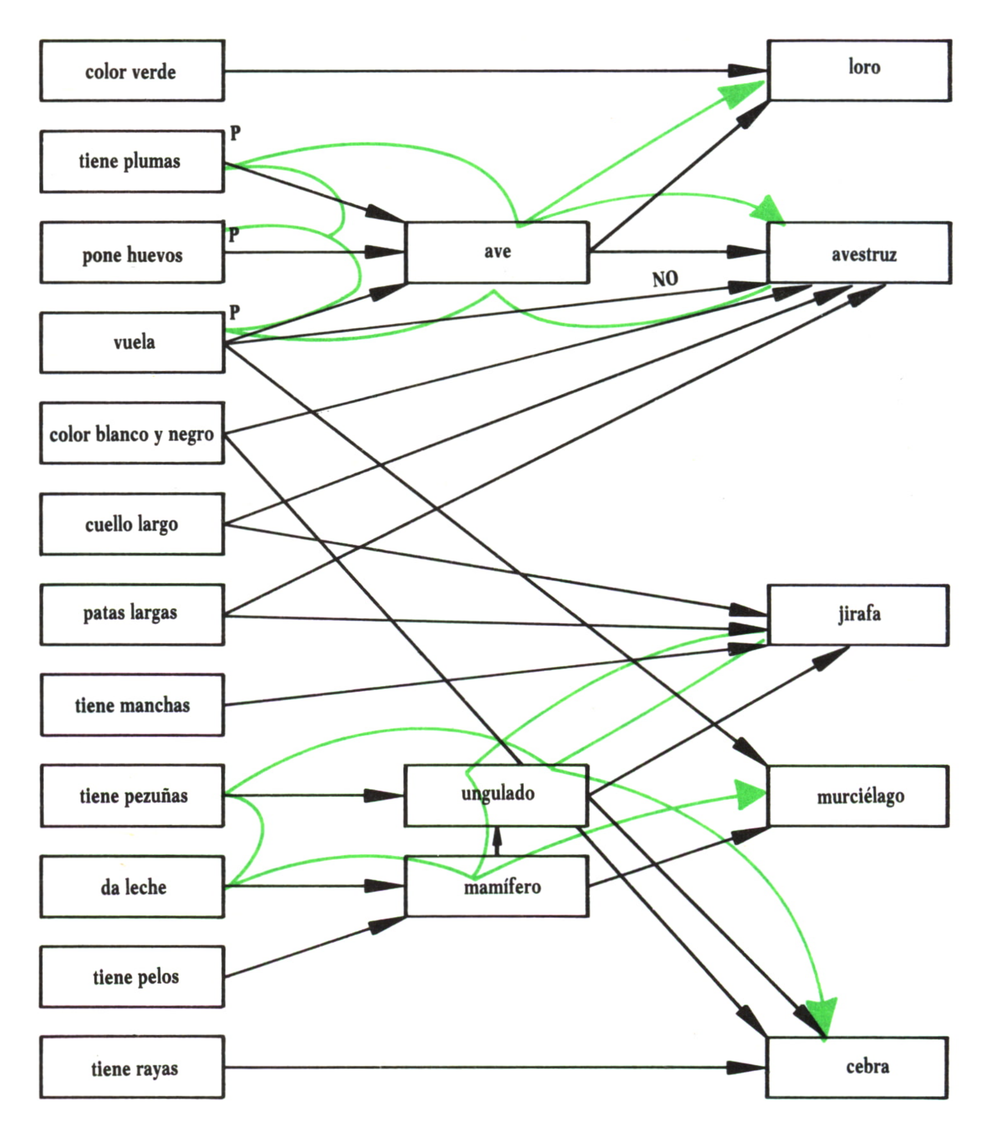


Fig. 12. Búsqueda en la base de conocimientos.

mente, si «es girafa», ni si «es cebra». El proceso concluye, pues no se puede obtener mas información.

La regla 2 no ha sido utilizada por inutil (su conclusión ya ha sido demostrada a partir de la regla 1 cuando contestamos afirmativamente a la pregunta de si «da leche»).

El proceso que hemos descrito en este tercer caso se suele llamar de

«encadenamiento mixto» y cada dia está mas extendido en los sistemas expertos y herramientas que utilizan reglas de producción.



Búsquedas en profundidad y en anchura

En la búsqueda de nuevos hechos a través de la base de conocimientos podemos «desplazarnos» hacia atrás o hacia adelante, como hemos visto. Pero, por otro lado, hay que tener en cuenta otra posibilidad. En la búsqueda con encadenamiento hacia atrás se partía de un objetivo para averiguar los hechos de que hay que disponer para poderlo demostrar. Hemos tomado el primero de estos hechos como nuevo subobjetivo y así hemos llegado hasta los datos de base necesarios (ver Fig. 12). Sólo después de llegar hasta un hecho de base hemos realizado el encadenamiento hacia adelante para volver después a considerar el siguiente subobjetivo. Este esquema de análisis del conocimiento se llama búsqueda en profundidad.

Pero no es éste el único camino posible; en efecto, podíamos haber examinado todos los objetivos cuya demostrabilidad tratamos de establecer y haber visto de qué hechos hay que partir para tomarlos como subobjetivos; analizar, a continuación, estos subobjetivos para saber que hechos hay que demostrar, etc. En la figura 13 se muestra gráficamente como podría ser este otro camino. A este esquema de análisis del conocimiento se le llama búsqueda en anchura o en amplitud.

Naturalmente, se puede programar un proceso de búsqueda en anchura junto con un esquema de encadenamiento hacia adelante.

En general se puede decir que si el número de hechos de partida no es muy numeroso pero los hechos a demostrar sí lo son, una búsqueda hacia adelante será más adecuada; si, por el contrario, se dispone de un solo objetivo (o varios, pocos) y los datos (hechos) de los que se dispone o puede disponer son numerosos, será más eficaz el encadenamiento hacia atrás.



«Backtracking»

Se suele utilizar este término inglés para designar un proceso de «vuelta atrás» que se realiza con cierta frecuencia en los procesos de búsqueda. En efecto, el esquema de examen de la base de conocimiento no suele llevar un camino en una sola dirección, sino que, en ocasiones, llega a un punto en que debe «volver atrás» y seguir por otro camino ó bién detener su búsqueda.

Hemos visto en el ejemplo expuesto anteriormente (y gráficamente también en la figura 12) cómo al llegar a examinar si es demostrable que el animal en cuestión «es ave» y no disponer del dato de si vuela o no, después de averiguarlo (preguntándolo al usuario) hemos vuelto hacia atrás

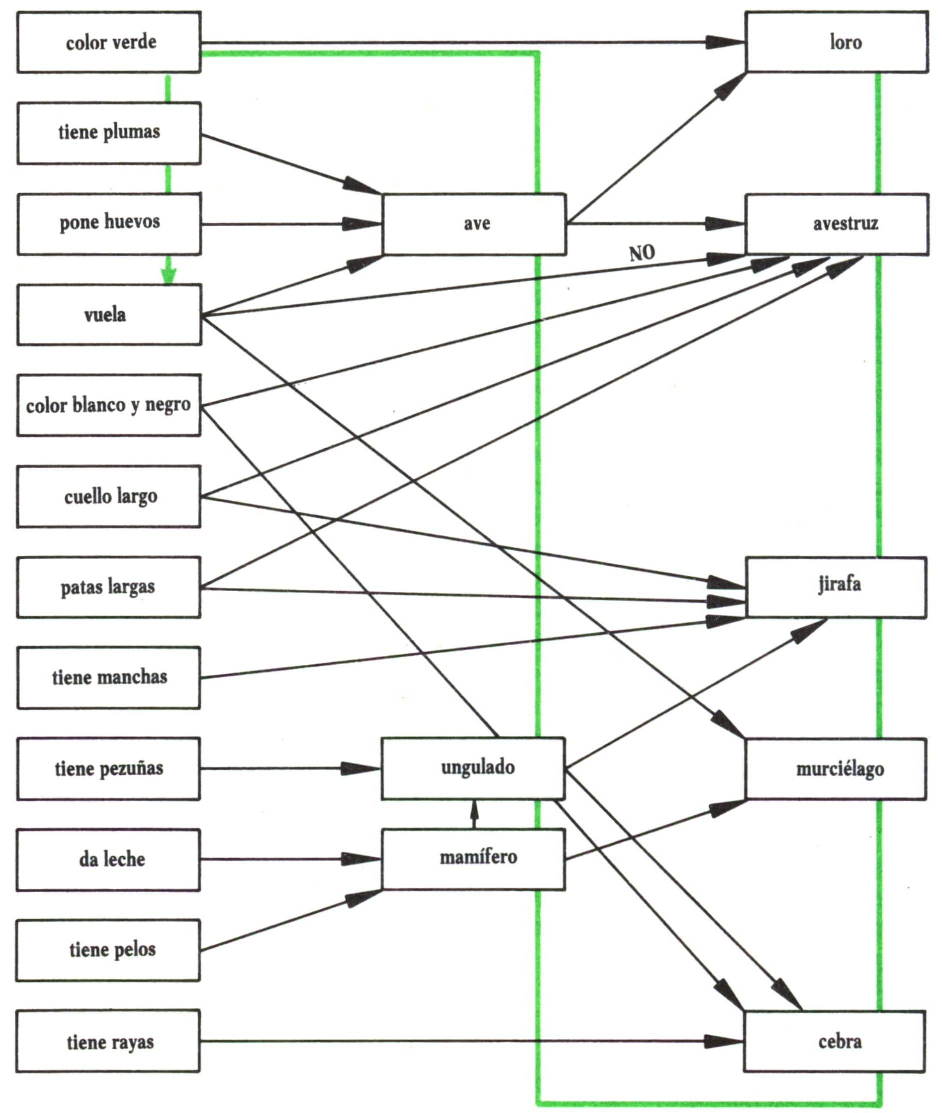


Fig. 13. Búsqueda en anchura con encadenamiento hacia atrás.

para considerar otra de las características necesarias («poner huevos») y, como la contestación ha sido «NO», hemos vuelto a considerar una segunda vez la cualidad de ser ave, ahora con la regla 4, para preguntarnos si «tiene plumas». Del mismo modo, examinando el objetivo «girafa» hemos seguido el camino «ungulado» —«mamífero»— «da leche» y, después de sacar conclusiones, hemos hecho «backtracking» para volver desde «mamí-

fero» a «ungulado» y seguir en profundidad con la línea «tiene pezuñas».

El proceso de «vuelta atrás» es sumamente interesante y añade mucha flexibilidad al esquema de búsqueda, pero requiere un conjunto adicional de controles que dificultan la programación del motor de inferencia (hay que guardar información sobre los valores de los hechos en cada paso para retomar la situación —hechos demostrados y no demostrados, reglas utilizadas y reglas de las que no se ha hecho uso, etc.— en el punto a donde se vuelve para tomar un camino alternativo).



Utilización de lógica de primer orden

Si en lugar de utilizar lógica de proposiciones se hace más sofisticado el entorno de representación del conocimiento pasando a utilizar lógica de primer orden (cálculo con predicados) la representación de datos es más flexible, pero el control del proceso resulta más complicado. En efecto, la introducción de variables en las reglas de producción supone abordar un verdadero riesgo de «explosión» de hechos demostrados o por demostrar. En los ejemplos vistos hasta ahora las proposiciones (hechos) ha manejar estaban dados, y solo dudábamos de su valor («el animal vuela» es verdadero=V, falso=F o desconocido=D); sin embargo, si decimos «todo ungulado es un mamífero» o, lo que es lo mismo, «si X es ungulado, X es mamífero» (es decir «ungulado(X) mamífero(X)») para sacar conclusiones, debemos «instanciar» o particularizar la variable «X» (en principio, cualquier animal del pequeño zoo que estamos manejando): y así preguntarnos si será verdad o nó que ungulado(loro), ungulado(girafa), ungulado(avestruz), ungulado(murciélago) y ungulado(cebra). Además, como, normalmente no se producirá la instanciación de la regla para todos los valores posibles, sino sólo para aquél o aquéllos que en cada momento nos interesa, las reglas ya no son (como en el caso del cálculo de proposiciones) de utilización una única vez, sino que en cada paso deductivo se pueden utilizar, con lo que hay que controlar no si la regla se ha utilizado o nó, sino si ha sido usada con tal o cual variable.

Además, en cada regla puede haber varias variables que se particularizan independientemente unas de otras. Concretamente, si en una regla aparecen cuatro variables (caso usual en un sistema real) y disponemos de cincuenta hechos conocidos en el momento considerado, se pueden probar 50 elevado a 4 (más de 6 millones) de combinaciones diferentes: ya se comprende que, por muy potente que sea el equipo sobre el que se esté trabajando, la búsqueda exhaustiva es impensable en un caso como el considerado. Es necesario establecer un mecanismo automático de conducción del proceso deductivo. En cada momento, el orden (dinámico) en que conviene evaluar las premisas para localizar el conjunto de valores (cuatro en el caso que acabamos de proponer) que hay que seleccionar

para aplicar la regla correspondiente, depende del contexto y/o de la propia extructura de la regla.

El ciclo de base de un motor de inferencia suele tener tres fases: localización de las reglas utilizables en ese momento, elección de la regla a aplicar entre las posibles (en el caso de la lógica de predicados que estamos considerando es el momento de instanciación de las variables para elegir no sólo la regla a utilizar sino la n-tupla de valores con los que se va a usar) y realización de la deducción propiamente dicha con la regla y valores considerados.

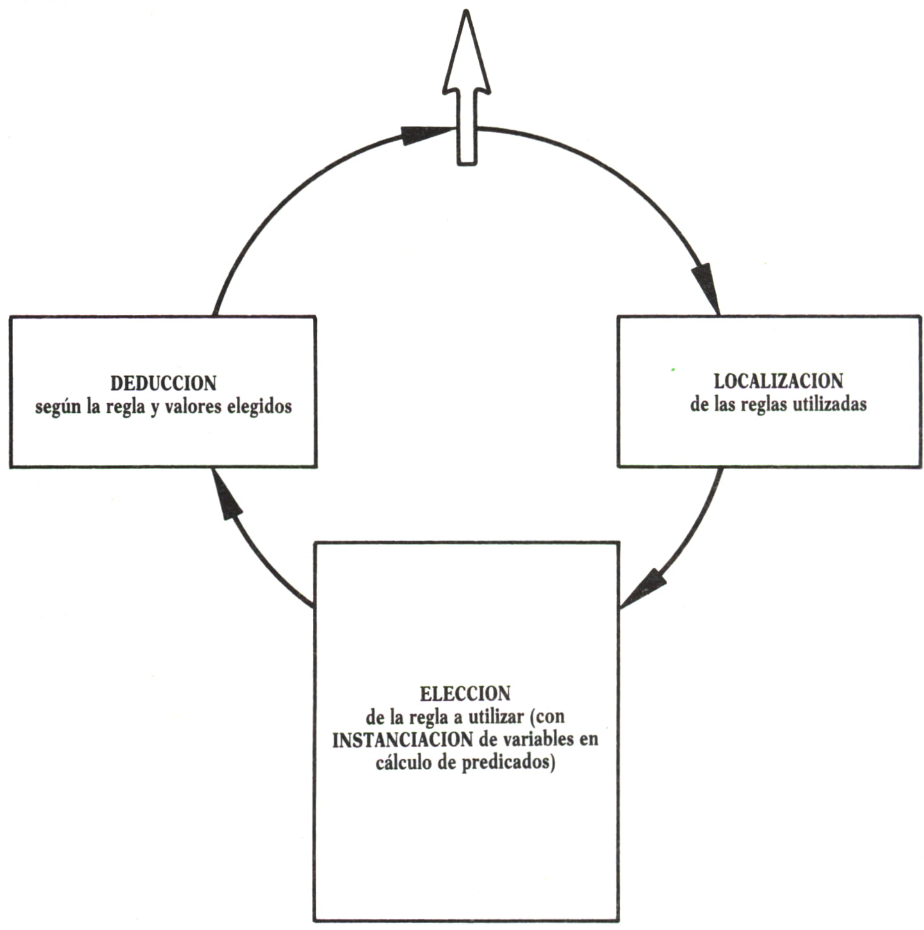


Fig. 14. Ciclo de base del motor de inferencia.

Las estructuras de control utilizadas en los diferentes sistemas son muy diferentes (elección en función del interés de la regla según el objetivo a

obtener, elección según un juego de hechos considerados básicos, elección de la primera regla encontrada, búsqueda exhaustiva, etc.).

Uno de los sistemás mas interesantes es el utilizado en el generador de sistemas expertos SNARK: en la etapa primera de detección de las reglas a utilizar se considera para cada regla un conjunto de «palabras clave» que son las *propiedades* o *valores* significativos que aparecen en la regla y se marca la regla como inutilizable (en ese momento) si alguna de sus palabras-clave no aparece en la base de hechos; si, por el contrario, todas las palabras-clave de la regla aparecen al menos una vez en la base de hechos se anota que esa regla es utilizable en ese ciclo.

A continuación se van examinando las reglas «utilizables», para la localización de la «n-tupla» de valores que permitirán activar esa regla.

Para ello, considera tres tipos diferentes de premisas y selecciona la regla según el tipo de premisas que aparecen en cada una.

Los tres tipos son:

tipo 1. «propiedad N(X)=a», y la instanciación consiste en localizar un elemento incógnito «X» que cumpla que su «propiedad N» valga «a». Este elemento «X» (candidato provisional a ser uno de los valores seleccionados) se llama «zombi», en el argot.

tipo 2. «propiedad N(Z)=X», donde «Z» es un elemento (zombi) resultado de una instanciación anterior. X es el nuevo zombi a localizar.

tipo 3. «propiedad N(X)=Y», hay que localizar simultáneamente una pareja de valores que cumpla la expresión.

Se selecciona la regla según el tipo de premisas que aparecen en ella (primero tipo 1, luego tipo 2, por último el tipo 3) y, en caso de igualdad de tipo de premisas, se elige aquella cuya propiedad aparece mayor número de veces en la base de hechos.

Se recorre la base de hechos seleccionando (encadenando) por *valor* o *propiedad* y se toman los objetos que se adecúan al modelo de búsqueda que estemos realizando (se va creando una pila con los valores obtenidos). A continuación se «repercuten» los valores seleccionados sobre las premisas no evaluadas y sobre la parte de conclusión de la regla. De esta implicación pueden surgir instanciaciones nuevas o cambio de tipo de premisas, que son tomados en cuenta en el siguiente ciclo.

Si aparece una contradicción con la base de hechos (que se ha utilizado) o no se ha podido realizar elección alguna de regla, se vuelve atrás, se saca de la pila la anterior selección hecha y se continúa el proceso con ese nuevo valor.

Naturalmente, la estrategia descrita es un caso particular (interesante, eso sí) de un motor de inferencia (cualquiera definido sobre un paquete SNARK) y las posibilidades que aparecen en distintos sistemas expertos son muy variadas. Hemos puesto este ejemplo para mostrar al lector cómo

se complica la gestión del proceso deductivo con la introducción de la lógica de primer orden.



Aplicación a otros esquemas de representación

Aunque el esquema de representación no sea la lógica, los sistemas de producción siguen siendo un procedimiento válido de deducción. Podemos pensar en un esquema de representación de tipo atrubuto-valor para el objeto: «ir al teatro» (*). Se utilizarán las parejas correspondientes a los siguientes atributos: «Distancia» (para representar la distancia en Km de mi casa al teatro), «Tiempo» (tiempo de que disponemos hasta el comienzo de la representación, en minutos), «Medios» (procedimiento genérico para ir al teatro; puede ser conducir o andar), «Situación» (se refiere a la ubicación de mi casa; puede ser: afueras o centro), «Meteorología» (que representa el tiempo atmosférico que hace; puede ser: bueno o malo) y «Acción» (modo efectivo por el que iré al teatro; se consideran cuatro posibles acciones a tomar: pedir-un-taxi, ir-en- coche, abrigarse-e-ir-andando e ir-andando).

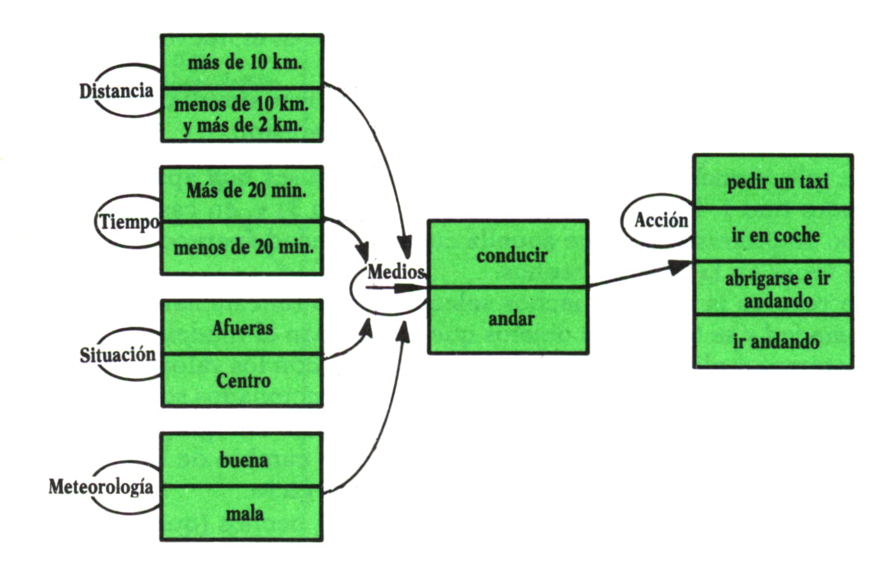


Fig. 15. Representación mediante parejas atributo-valor.

(*) Seguimos un ejemplo presentado por Harmon y King en su libro *Expert Systems*. Ed. John Wiley & Sons. 1985.

Partimos de la siguiente base de conocimientos:

Regla 1. Si Distancia es más de 10 Km → Medios es conducir
Regla 2. Si Distancia es más de 2 Km pero menos de 10 Km
y Tiempo es menos de 20 min → Medios es conducir
Regla 3. Si Distancia es más de 2 Km pero menos de 10 Km
y Tiempo es más de 20 min → Medios es andar
Regla 4. Si Medios es conducir y
Situación es afueras → Acción es pedir-un-taxi
Regla 5. Si Medios es conducir y
Situación es centro → Acción es ir-en-coche
Regla 6. Si Medios es andar y
Meteorología es mala → Acción es abrigarse-e-ir-andando
Regla 7. Si Medios es andar y
Meteorología es buena → Acción es ir-andando

Si tratamos de averiguar «cómo ir al teatro» el objetivo es saber qué valor debe tomar el atributo Acción. Se puede comenzar, con encadenamiento hacia atrás, viendo que el objetivo Acción (con diferentes valores asociados) aparece como consecuente en las reglas 4, 5, 6 y 7.

Para saber si se puede deducir el valor «pedir-un-taxi» del atributo Acción hay que saber si Medios tiene el valor «conducir» (o no) y si situación tiene el valor «afueras» o no.

Para averiguar el valor de Medios debemos acudir a las reglas 1, 2 y 3 que tienen este atributo como consecuente. De la regla primera obtenemos la necesidad de saber la distancia a que se encuentra el teatro de nuestra casa: el sistema lo preguntará y si la respuesta es 5 Km, por ejemplo, es decir «menos de 10 Km pero mas de 2 Km» no se puede activar la regla 1. Toma la siguiente posibilidad de obtener un valor para el atributo Medios (objetivo de búsqueda actual). La regla 2 le «sugiere» preguntar por el tiempo de que disponemos para ir al teatro. Si contestamos que tenemos 10 min. de tiempo, se dispara la regla 2 y pasamos a saber que Medios vale «conducir».

Dependiendo de como esté previsto el proceso de control de deducciones, ahora se aplicará el dato obtenido, en encadenamiento hacia adelante, para llegar a concluir valores de Acción o no; y, por otro lado, seguirá examinando las restantes posibilidades del atributo Medios (según la regla 3) o no; puede que se admitan varios valores de un atributo (en este caso significarían opciones alternativas para ir al teatro) o que se consideren contradicciones y se rechacen.

Una vez sabido que Medios es «conducir», el sistema preguntará la situación de la casa (regla 4). Si decimos que «afueras», deducirá el valor

«pedir-un-taxi» para Acción. Si solo queríamos un valor (o el sistema está preparado para dar un solo valor de cada atributo) el proceso deductivo concluye. En caso contrario, el proceso continúa observando otros posibles valores del atributo Acción, pero las reglas restantes (5, 6 y 7) no pueden activarse (por la situación la 5 y por los medios la 6 y la 7). El proceso concluye.

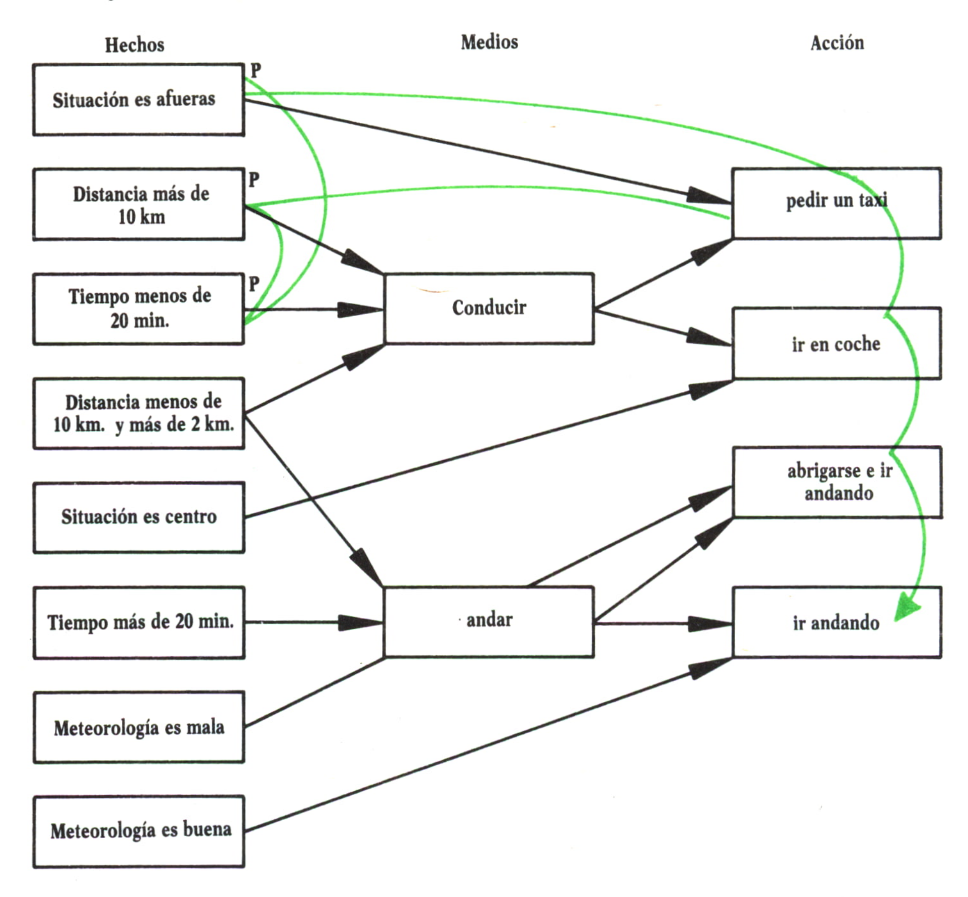


Fig. 16.

Como se ve, el modo de trabajo es el mismo ya descrito anteriormente, pero aquí se maneja globalmente el concepto de atributo (y se examinan sus posibles valores). Esto suele suponer el análisis global (aunque sucesivo, naturalmente) de varias reglas y suele dirigir el backtracking de tal modo que la búsqueda se circunscribe a los valores de un atributo antes de pasar a otro atributo o a otra regla genérica.



Otros procedimientos de control

Los esquemas de búsqueda descritos, representan procedimientos generales de gestión del proceso de inferencia en un sistema experto. Sin embargo, normalmente todo sistema concreto (o cualquier «herramienta» de generación de sistemas) incluye algún procedimiento adicional de control. Vamos a ver algunos de estos sistemas.

Hay dos modos distintos de introducir un control adicional: mediante instrucciones y parámetros de control o a través de metareglas. El primer modo consiste en añadir alguna instrucción que dirige el proceso de un modo concreto. El segundo, se realiza introduciendo en la base de conocimiento ciertas reglas de producción cuyo procesamiento produce que se dirija la búsqueda por un camino o por otro.

Dentro del primer grupo podemos citar los siguientes mecanismos: restricciones en el uso de ciertas reglas, estrategias alternativas de búsqueda, introducción de hechos ficticios, órdenes de parada o pausa, etc. En cuanto a las «metareglas» se puede hablar de reglas de «ver primero», de activación de paquetes de reglas, de manejo de contenidos, etc.

En algunos sistemas se puede introducir un parámetro en cualquier regla que indique al sistema que dicha regla *sólo puede ser utilizada en encadenamiento hacia adelante* (o hacia atrás): entonces el motor de inferencia despreciará esa regla cuando esté realizando un encadenamiento hacia atrás (hacia adelante, respectivamente).

Otros sistemas disponen de varias *estrategias típicas que pueden ser elegidas alternativamente*. Por ejemplo: la estrategia consistente en que «la regla primera a elegir es aquella que tiene el menor número de condiciones todavia sin concretar», se llama MINCOND; «elegir la regla que tiene entre sus premisas el hecho que hayamos establecido mas recientemente» es HECHREC; «tomar la regla que tiene mas conclusiones» es MAXCONC; etc. De tal modo que podemos realizar dos tipos de control (según nos lo permita el generador que estemos utilizando o según lo programemos): detener el proceso en un momento dado e introducir la orden correspondiente desde el exterior del sistema o bien incluir en la base de conocimientos una regla que contenga esa información: «si da-leche y es un ungulado entonces HECHREC». Cuando se active esta regla el motor de inferencia entiende la orden HECHREC y modifica la estrategia a aplicar a partir de entonces.

En ocasiones, se pueden introducir hechos ficticios para guiar el proceso de un modo diferente al previsto. Imaginemos que nuestro motor tiene previsto (como estrategia de búsqueda) seleccionar ante todo aquella

regla, de entre las posibles, que tenga menos premisas (menos condiciones en el antecedente). Si disponemos de las reglas

R1: Si vuela y
pone huevos → es un ave
R2: Si tiene plumas → es un ave

el sistema seleccionará la segunda y después la primera. Si deseamos, por alguna razón, que sean activadas en el orden en que aparecen, se puede modificar la base de conocimientos poniendo

R1: Si vuela y
pone huevos → conclusión-1
RF1: Si conclusión-1 → es un ave
R2: Si tiene plumas → conclusión-2
RF2: Si conclusión-2 → es un ave

el resultado será que al buscar en encadenamiento hacia atrás una regla que demuestre «es un ave», seleccionará ante todo la regla RF1 y para obtener su condición («conclusión-1») realizará R1. Sólo después irá a examinar RF2 y R2. Hemos cambiado el orden previsto originalmente en el motor de inferencia.

Del mismo modo, hay sistemas que disponen de la *orden PARADA* o *PAUSA* que puede aparecer en cualquier regla de producción o como comando en el sistema.

Por otro lado, se pueden introducir en el sistema *«metareglas»* o reglas con información sobre el proceso de las reglas. De este tipo son las *reglas de «ver primero»;* es decir, reglas del tipo «SI condición-1 y condición-2 ENTONCES ver-primero hecho-1 y hecho-2». Con esta regla se consigue cambiar el orden de búsqueda, dirigiendo el proceso hacia «hecho-1» y «hecho-2» en vez de seguir el proceso general previsto por el motor.

En ocasiones se introducen *reglas de activación de un paquete de reglas.* Si hay un grupo de reglas (un paquete) que sólo queremos que se active en unas condiciones determinadas, se puede incluir una condición fija en las premisas («condición-de-paquete-1», por ejemplo) y cuando se dispare la regla dada (que comporta entre sus conclusiones la «condición-de-paquete-1») se pondrán en situación de ser activadas todas las reglas del paquete (todas las que tengan como premisa la tan repetida «condición-

de-paquete-1»). Todo lo dicho sirve para encadenamiento hacia adelante. Si se está trabajando en encadenamiento hacia atrás la regla de activación del paquete deberá tener la clave entre sus premisas y el paquete deberá tener en sus conclusiones dicha clave.

Por último, vamos a comentar que, en ocasiones, las metareglas aluden no a la posición o tipo de las reglas de que se trata sino que dirigen el funcionamiento del motor *aludiendo a las reglas por su contenido.* Por ejemplo, en el clásico MYCIN aparecen metareglas como las siguientes

R1. Si se busca una terapia,
ENTONCES considerar, por el orden dado, las reglas que permiten

1. adquirir las informaciones clínicas sobre el paciente;
2. descubrir qué organismos, si hay alguno, son la causa de la infección;
3. identificar los organismos más probables;
4. descubrir los medicamentos potencialmente útilies en el caso;
5. elegir los más adecuados y en menor número.

R2. Si 1. el paciente es sujeto de alto riesgo;
2. existen reglas que mencionan las pseudomanías en algunas de sus premisas;
3. existen reglas que mencionan las Klebsiellas en alguna de sus premisas,
ENTONCES es probable que sea preferible utilizar antes las del tipo aludido en 2, que las aludidas en 3.

Hay dos elementos de conocimiento muy utiles, y representativos en numerosas ocasiones de las situaciones reales, a los que aún no hemos aludido aún; queremos hacerlo ahora como conclusión de este apartado; son el proceso de la incertidumbre y el proceso no-monótono.



Razonamiento aproximado

Se alude con este nombre al proceso de los datos de incertidumbre que podemos (o debemos, en ocasiones) introducir en los sistemas.

En efecto, hay muchos casos en que al formalizar una aseveración (escribir una regla de producción) o bién al establecer un hecho, no se tiene

la plena seguridad de que sea cierto lo que estamos afirmando. Esta circunstancia se suele incluir en la base de conocimiento mediante un factor de incertidumbre o de certeza. Puede aparecer a nivel del experto mediante la introducción del factor correspondiente en la regla o bien a nivel del usuario como un factor que afecta a los hechos que se definen como falsos o verdaderos.

Un ejemplo del primer caso seria la regla

> «si el cultivo es sangre y la forma del microorganismo es bastón entonces es probable (0,6) que el organismo sea un pseudomonas aeruginosa»

Ya hemos aludido a ello anteriormente, así como a las escalas (de −1 a +1 o de −100 a +100) que se suelen utilizar. Aquí, comentando la estructura del motor de inferencia, hemos de aludir a cómo se manejan estos datos en el proceso deductivo.

Existen numerosos procedimientos de gestionar los factores de certeza según la finalidad del sistema experto de que se trate, pero los elementos que se consideran, básicamente, son dos: umbral de activación de una regla y «propagación» de la incertidumbre en los procesos deductivos.

Cuando no se realiza razonamineto impreciso o aproximado, una regla se activa si son ciertas todas las premisas. Cuando los hechos se conocen, por el contrario, afectados de un factor de fiabilidad («es cierto que el animal correspondiente da leche —coeficiente 0,7—») hay que establecer un límite inferior (umbral) a partir del cual se considera que los hechos son «suficientemente ciertos» como para activar la regla correspondiente. Es usual que haya disponibles instrucciones para modificar ese umbral bien de un modo general o a través de metareglas.

Por otro lado, el motor de inferencia debe tener establecidos unos mecanismos de cálculo de los coeficientes de fiabilidad para los hechos deducidos en función de los coeficientes de las premisas y/o del coeficiente general de la regla de producción correspondiente. El mecanismo más utilizado en la mayoría de los sistemas es el de «propagación multiplicativa con acumulación de plausibilidad (o certeza)». Ya en MYCIN se utilizó este procedimiento, que ha aparecido posteriormente en numerosos sistemas, con un esquema semejante al siguiente:

a) Combinación de certezas para una sola regla que concluye en un hecho:

a.1.) si la conclusión no tiene coeficiente;

hecho-1 y hecho-2 → hecho-3
(cf=0,5) (cf=0,3) (cf=0,3)

(coeficiente de la conclusión igual al mínimo de los coeficientes de las premisas.)

a.2.) si la conclusión tiene coeficiente de certeza;

hecho-1 y hecho-2 → hecho-3 (con factor 0,5)
(cf=0,5) (cf=0,3) (cf=0,15)

(coeficiente de la conclusión igual al mínimo de los coeficientes de las premisas (0,3) multiplicado por el coeficiente de la regla (0,5).) (Esta es la propagación mutiplicativa.)

b) Acumulación de plausibilidad cuando varias reglas concluyen sobre un mismo hecho:

Si la regla R1 concluye hecho-3 con coeficiente cf=0,3
y la regla R2 concluye hecho-3 con coeficiente cf=0,2

se produce una «acumulación» de la certeza sobre ese hecho-3 («va apareciendo» progresivamente como mas cierto cada vez). El factor acumulado que se atribuye al hecho-3 será de 0,44 según la siguiente regla:

$$cf_c = cf_1 + cf_2 - cf_1 * cf_2$$
(si cf_1 y cf_2 son positivos)

(donde cf_1 es el coeficiente de certeza de C cuando se obtiene como conclusión de la R1; cf_2 el correspondiente valor de C obtenido a partir de la R2 y cf_c el coeficiente acumulado que se calcula)

$$cf_c = cf_1 + cf_2 + cf_1 * cf_2$$
(si son ambos negativos)

y $cf_c = cf_1 + cf_2 / (1-\min) (|cf_1|, |cf_2|))$
(si son de diferente signo)

(siendo $\min(|cf_1|, |cf_2|)$ el menor de los valores absolutos de cf_1 y cf_2)

Esta regla se puede justificar de un modo gráfico diciendo que si tenemos una certeza de 0,3 para el hecho C (a partir de la regla 1) y se nos ofrece una plausibilidad

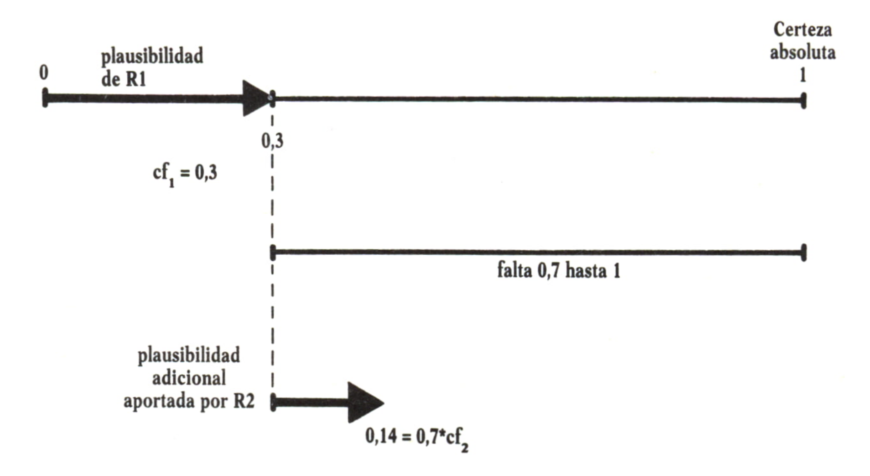


Fig. 17. «Acumulación de certeza» sobre un hecho.

adicional (de la regla R2) de 0,2, se incrementa la certeza que teníamos «aproximándola» hacia la certeza absoluta (valor cf=1) el 20% de la cantidad que le «faltaba» para llegar a 1 (en nuestro caso «faltaba» 0,7 y, por tanto, se produce un incremento de certeza de 0,14.



Procesos no monótonos

Los procesos deductivos que hemos considerado hasta ahora se han realizado aceptando que en cada «sesión de trabajo» los hechos eran verdaderos o falsos de un modo estable, de manera que si llegabamos a concluir su valor, se fijaban en ese valor y no se volvian a modificar (incluso se «tachaban» todas las reglas correspondientes). Este tipo de razonamiento se llama «monótono», pues de hecho, la base de conocimientos crece de un modo permanente («monótono») y nunca decrece.

Sin embargo, no repugna la idea de obtener provisionalmente que «el animal es un ungulado» (con alguna finalidad demostrativa concreta) para llegar a la conclusión, posteriormente, de que no lo es. Parece que se adecúa más a la relaidad de la actividad deductiva esta flexibilidad: es decir, preferiríamos que nuestro motor de inferencia fuera no-monótono. (En los procesos de establecimientos de planes, por ejemplo, esta capacidad de elaborar hipótesis que luego son evaluadas y, eventualmente, rechazadas es básica.)

De hecho, ya existen sistemas que aceptan esta posibilidad (con algunas restricciones) y sería deseable poder trabajar siempre en esta línea; pero las dificultades de manejo del conocimiento en los sistemas no-monótonos es grande y la mayoría de los sistemas trabajan de modo monótono exclusivamente.

CONSTRUCCION DE SISTEMAS EXPERTOS





OMO ya comentamos en el capítulo 1, la tarea de implementar un SE es una actividad propia de un profesional (un «ingeniero del conocimiento») y la discusión pormenorizada de las etapas a cubrir, las dificultades que pueden surgir, su solución, etc. rebasa el tamaño y finalidad de este libro. Entendemos, sin embargo, que puede ser de utilidad al lector la información que incluimos a continuación en dos apartados diferentes: por un lado presentamos un panorama genérico de la puesta en marcha de un sistema y por otro comentamos las diferentes herramientas comerciales disponibles en el mercado (especialmente para ordenadores pequeños).



IMPLEMENTACION DE UN SISTEMA EXPERTO

En el desarrollo e instalación de un SE intervienen un conjunto de personas que, dependiendo del tamaño del proyecto y del contexto en que se vaya a realizar, varían enormemente. Sin embargo, en la mayoría de los casos suelen estar presentes los siguientes grupos o personas:

a) Un responsable del proyecto; persona encargada del encuadramiento total del sistema y de supervisar la marcha general de los trabajos;

b) un equipo responsable del desarrollo: normalmente constituido por un ingeniero del conocimiento (cuya misión es favorecer la expresión y formalización de los conocimientos del experto, asegurando la concepción y arquitectura del sistema) y una o varias personas más, técnicos en inteligencia artificial y en informática clásica;

c) un experto en la materia propia del trabajo a realizar, encargado de aportar las informaciones necesarias para constituir la base de conocimientos inicial del sistema;

d) los utilizadores o usuarios que, normalmente, intervienen en la etapa final de puesta a punto del sistema y de complementación y contraste de las informaciones incluidas en la base de conocimientos.

Por otro lado, las etapas que suelen seguirse hasta la instalación definitiva de un SE suelen ser (ver Grundstein y De Bonniéres, 1987):

1. Gestación del proyecto.

Normalmente, no existe una experiencia amplia sobre los problemas a resolver con la instalación de un sistema experto en una determinada Organización o Area concreta de ella.

Suele desarrollarse esta etapa preliminar mediante reuniones con las personas implicadas. A veces se incluye en un estado tan incipiente el análisis concreto de algún problema puntual mediante la utilización de alguna herramienta adecuada, para evaluar posibilidades, obtener primeras conclusiones y gestar «ideas utilizables». En ocasiones se llama a esta etapa «de fertilización», por analogia con dicha actividad agraria.

2. Selección del problema.

Normalmente, mediante el proceso de «fertilización» o «sensibilización» anterior suelen surgir varias actividades que pueden ser abordadas. Es necesario seleccionar una.

A poco importante que se prevea el proyecto a desarrollar, suele realizarse un estudio de oportunidad y de impacto en la Organización y/o en la estructura informática existente (si hay alguna en ese área).

El proceso de selección se realiza, habitualmente, por «aproximaciones sucesivas» reteniendo las tareas que más se adecuen a las características típicas que «debe tener» un problema para ser resuelto mediante un SE, como ya se comentó en su momento.

3. Identificación del problema y evaluación del proyecto.

Una vez seleccionado el problema globalmente, es necesario definir sus perfiles, especificar las funciones que ha de realizar, y las características que ha de tener para responder a las necesidades expresadas. En esta fase hay que cubrir específicamente cuatro actividades:

— caracterizar el tipo de problema y la naturaleza de los conocimientos útiles para su resolución;
— especificar los modos de representación mas adaptados al caso;
— determinar y evaluar las «herramientas» que permitan implementar el esquema de representación elegido de un modo mas eficaz; y
— estimar el volumen de conocimientos a manipular, prever la dificultad que puede haber en explicitarlos y deducir los medios y el tiempo necesarios para la implementación.

4. Concepción y realización de un prototipo.

El desarrollo del prototipo es básico para poder evaluar la validez de las elecciones hechas en la etapa anterior. En la realidad suele suceder que el modo implementado la primera vez resulta bastante válido como solución final (sobre todo si se han desarrollado cuidadosamente las etapas anteriores), por lo que ésta es una de las fases más importantes de todo el proyecto, desde el punto de vista de la ingeniería del conocimiento.

El desarrollo del prototipo se suele hacer en un proceso que podriamos llamar cíclico, pues una vez concebido, se construye de acuerdo con el modelo diseñado, se prueba y compara, para deducir nuevos datos de diseño que conducen a un refinamiento del modelo concebido y seguir con el proceso cíclico hasta concluir un prototipo satisfactorio.

5. Complementación del prototipo y realización del sistema completo.

Una vez establecido a satisfacción un prototipo válido, debe desarrollarse (complementando la base de conocimientos) y añadiendo, quizá, módulos adicionales. Esta es una tarea larga (costosa en hombres y tiempo) pero más convencional que las anteriores.

6. Instalación en su ambiente-natural.

El paso de las fases anteriores (más de laboratorio) a la instalación efectiva del sistema, implica a elementos de la Organización que no han intervenido hasta el momento en el proyecto (o que lo han hecho de un modo superficial), por lo que es una etapa delicada. El grupo destinatario del sistema debe «hacerlo suyo», lo que comporta algunas actividades que hay que realizar:

— formalización e introducción de conocimientos nuevos;
— adaptación del vocabulario de la base y adaptación a los lenguajes de los usuarios;
— reflexión general sobre el impacto que el nuevo sistema tendrá en la Organización;
— formación de los usuarios.

7. Explotación y evolución del sistema.



HERRAMIENTAS DE DESARROLLO

Como ya hemos comentado, el desarrollo de sistemas expertos a partir de los lenguajes de programación existentes es una tarea complicada (para un no experto), especialmente en la parte de control del motor de inferencia y supervisión de las tareas generales.

Sin embargo, existen numerosas herramientas (paquetes de herramien-

tas) que facilitan esta tarea. Históricamente, primero se obtuvieron una serie de herramientas de generación de sistemas expertos a partir de los propios sistemas construidos (la más antigua de todas es EMYCIN, obtenida «vaciando» el sistema MYCIN de su base de conocimientos); posteriormente se ampliaron y mejoraron estos «sistemas vacios» o «esqueletos» con otras herramientas auxiliares (módulos de seguimiento, de control del conocimiento que se va incorporando, facilidades gráficas, etc.); por último, se han construido sistemas mixtos que incluyen varios paradigmas de representación del conocimiento, que integran diversas técnicas e, incluso, lenguajes de programación: a estos sistemas se les denomina «sistemas concha» o «shell».

Se puede decir que si, mediante un lenguaje como el LISP de los primeros años 70, se podía tardar de dos mil a cinco mil horas x hombre en cada regla de un sistema basado en el conocimiento, en el año 1975 (con INTERLISP) se había reducido el coste hasta doscientas a quinientas horas x hombre: en nuestros días, con las herramientas disponibles, se puede establecer en cinco a ocho horas x hombre el coste de la programación de cada regla.

Presentamos a continuación un conjunto de estos paquetes de herramientas o sistemas concha para información del lector, habiendo elegido los más populares de entre los «grandes» (concebidos para trabajar en ordenadores de tamaño medio aunque muy potentes) y los más accesibles de entre los pequeños (algunos disponibles, incluso, para ordenadores personales).



Advise

Es un sistema experto de propósito general (un esqueleto) de ayuda en la construcción de sistemas. Consiste en un conjunto integrado de herramientas de desarrollo. Estas herramientas incluyen soportes para múltiples formas de representación del conocimiento (reglas, redes semánticas y bases de datos relacionales), soportes para varios esquemas de propagación de certezas (probabilidades, Bayesiano aproximado, lógica min/max, y acumulación de evidencia), soporte de varias estrategias de control (optimización por utilidad, búsqueda en amplitud según probabilidades, y encadenamiento hacia adelante y hacia atrás). También incorpora programas de aprendizaje inductivo (GEM y CLUSTER) para derivar de forma inductiva reglas de decisión e información a partir de los ejemplos. GEM generaliza ejemplos de distintos conceptos y crea reglas de lógica formal para reconocerlos. CLUSTER construye automáticamente una clasificación de entidades dadas utilizando una aproximación global al concepto. En el modo de explicación, ADVISE parafrasea (en inglés) las reglas de decisión, permite consultas simples de la base del conocimiento, y visualiza los pa-

sos de razonamiento. ADVISE está implementado en PASCAL y opera sobre ordenadores DEC VAX bajo un sistema operativo UNIX. ADVISE fue desarrollado en la Universidad de Illinois, como sistema de investigación.



Alice

Es un lenguaje propio de ingeniería del conocimiento concebido para representaciones basadas en la lógica. Sus características principales son un vocabulario extensivo para describir problemas combinatorios en investigación operativa y un proceso de consecución de objetivos que incluye la búsqueda de cualquier solución factible, construyendo repetidamente mejores soluciones, y dando además la prueba de optimización de la solución. ALICE fue desarrollado en el Instituto de Programación, en París, como sistema experimental. La palabra ALICE está formada por las iniciales de su nombre en inglés: A Languaje for Intelligent Combinatorial Exploration (Un lenguaje para la Exploración Combinatoria Inteligente).



AL/X

Es un lenguaje propio de ingeniería del conocimiento diseñado, en principio para representaciones basadas en reglas; pero también puede soportar métodos de representación basados en marcos. Sus principales características son una combinación de esquemas de control de encadenamiento hacia adelante y hacia atrás, una red semántica que relaciona los componentes de las reglas, mecanismos de manejo de certezas y consultas automáticas del usuario. El entorno de soporte contiene facilidades para explicar el razonamiento del sistema. El AL/X tiene muchas similitudes con el sistema KAS que está extraído de PROSPECTOR eliminando sus conocimientos sobre geología. El AL/X está implementado en PASCAL y opera sobre un PDP-11/34 con sistema operativo UNIX. Fue desarrollado por Intelligent Terminals, Ltd., como un sistema de investigación (Advice Languaje X).



Art

ART es un lenguaje de ingeniería del concimiento preparado para representaciones basadas en reglas. Además de las reglas, también soporta métodos de representación basados en marcos y representaciones orientadas a los procedimientos. Sus principales características incluyen esquemas de control de encadenamientos hacia adelante y hacia atras, mecanis-

mos de manejo de certezas y mundos hipotéticos (que son una forma de estructurar la base de datos definiendo los contextos en los que pueden aplicarse las reglas y los hechos). El entorno del soporte incluye ayudas de depuración standard, como por ejemplo facilidades de seguimiento de razonamiento (trace) y paquetes de interrupción del razonamiento, que funcionan en conjunción con un monitor gráfico. ART está escrito en LISP y opera sobre máquinas CADR y sobre el Symbolics 3600. Fue desarrollado por Inference Corporation como sistema comercial (Advanced Reasoning Tool).



Emycin

Es un esqueleto de sistema experto que se puede utilizar como lenguaje de generación de nuevos sistemas expertos, para representaciones basadas en reglas. Sus principales características incluyen esquemas de control de encadenamientos restrictivos hacia atrás (muy adecuado para problemas tipo consulta, mecanismos de manejo de certezas, y facilidades automáticas de consulta por el usuario). El entorno de soporte contiene facilidades de interfaz muy sofisticadas, que explican el razonamiento del sistema y sirven para adquirir nuevos conocimientos. El sistema está implementado en INTERLISP y fue diseñado originalmente para operar sobre un ordenador DEC PDP-10 bajo un sistema TENEX o TOPS20. EMYCIN fue desarrollado en la Universidad de Stanford, como sistema de investigación, y esencialmente consiste en el sistema experto MYCIN, al que se ha eliminado sus conocimientos sobre medicina (Essential MYCIN).



Expert

EXPERT es un esqueleto en forma de lenguaje de ingeniería del conocimiento concebido para representaciones basadas en reglas. Sus características principales son un esquema de control con encadenamiento hacia adelante diseñado especialmente para problemas de clasificación, mecanismos de manejo de certezas, y códigos muy eficaces y transportables. El entorno de soporte contiene facilidades de interface de usuario muy sofisticadas, que incluyen explicaciones, adquisición de conocimientos y comprobaciones de consistencia. EXPERT está implementado en FORTRAN, y funciona tanto sobre equipos DEC como IBM. Es uno de los lenguajes de ingeniería del conocimiento más conocidos y utilizados en todo el mundo de las aplicaciones médicas. EXPERT ha sido desarrollado en la Universidad de Rutgers, como sistema de investigación.



Expert-Ease

Es una herramienta para la construcción de sistemas que ayuda al experto en un dominio específico a construir un sistema experto. El experto define el problema en términos de características, o factores que llevan a resultados particulares. El sistema consulta al experto pidiéndole ejemplos que describan las condiciones que llevan a cada resultado. De los ejemplos, el sistema aprende un procedimiento para resolver el problema, y genera un árbol de decisión que representa ese procedimiento. EXPERT-EASE esta implementado en PASCAL, y opera sobre IBM PC o XT, con una memoria de 128 K. EXPERT-EASE ha sido desarrollado por Intelligent Terminals Ltd. de Gran Bretaña como sistema comercial.



Hearsay-III

HEARSAY-III es un lenguaje de ingeniería del conocimiento para representaciones basadas en reglas. Integra reglas con una arquitectura de «pizarra» (consistente en fuentes del conocimiento que se comunican a través de una pizarra central, o base de datos). Cada fuente del conocimiento es una colección de reglas que se ejecutan contrastando datos y conocimiento en la pizarra. En el HEARSAY-III la pizarra tiene dos partes: una pizarra de dominio (para el razonamiento sobre el dominio problema), y una pizarra de planificación (para el razonamiento sobre cuándo y cómo aplicar el conocimiento del dominio). HEARSAY-III soporta el diseño de sistemas que requieren procesamiento asíncrono de la información, y el diseño de estructuras de control para el manejo de objetivos múltiples. HEARSAY-III está implementado en LISP. Ha sido desarrollado por el Information Sciences Institute como sistema de investigación.



Kas

KAS es un lenguaje esqueleto de ingeniería del conocimiento para representaciones basadas en reglas. Básicamente se trata del PROSPECTOR eliminándole los comienzos sobre Geología. KAS utiliza reglas de inferencia con factores de certeza asociados, junto con una red semántica particional para codificar los conocimientos. Las inferencias están basadas en encadenamientos hacia adelante y hacia atrás, y la propagación de probabilidades a través de la red semántica. El entorno de soporte del KAS tiene facilidades para explicaciones y adquisición de conocimiento y dispone además de una facilidad de reconocimiento de sinónimos, revisión de respuestas, sumarización, y seguimiento. El sistema está implementado en IN-

TERLISP. Fue desarrollado por SRI International como sistema de investgación (Knowledge Acquisition System).



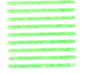
Kee

KEE es un lenguaje de ingeniería del conocimiento concebido para representaciones basadas en marcos. También soporta métodos de representación basados en reglas, orientados al procedimiento y orientados al objeto. Sus principales características son múltiples bases de conocimiento para facilitar el diseño modular del sistema y un intéprete de reglas con encadenamientos hacia adelante y hacia atrás. Su entorno de soporte incluye un paquete de depuración orientado a gráficos y unas facilidades de explicación que utilizan presentaciones gráficas para indicar las cadenas de inferencia. KEE está escrito en INTERLISP y opera sobre sistemas XEROX 1100 y Simbolics 3600. Fue desarrollado por Intellicorp para aplicaciones de genética molecular, pero, actualmente, puede adquirirse como sistema comercial de propósito general (Knowledge Engineering Environment).



Kes

Es un lenguaje de ingeniería del conocimiento para representaciones basadas en reglas y en marcos. Sus características principales son un esquema de control de encadenamiento hacia atrás, mecanismos de manejo de certezas, y un subsistema de clasificación de patrón estadístico basado en el teorema de Bayes. El entorno de soporte contiene facilidades de interface para adquirir nuevos conocimientos. KES está implementado en FRANZ LISP y opera sobre sistemas UNIVAC o VAX. Ha sido desarrollado por Software Architecture and Engineering, Inc., como sistema comercial (Knowledge Engineering System).



Loops

LOOPS es un lenguaje preparado para representaciones orientadas al objeto. También es capaz de soportar métodos de representación basados en reglas, orientadas al acceso y orientadas al procedimiento. Su característica principal es la integración de sus cuatro esquemas de programación, lo que le permite utilizar de un modo conjunto estos cuatro paradigmas en la construcción de sistema. Por ejemplo, las reglas y los conjuntos de reglas se consideran objetos LOOPS, y los procedimientos pueden ser funciones LISP o conjuntos de reglas. El sistema de soporte contiene herra-

mientas de depuración orientadas a la presentación, como paquetes de interrupción y editores. LOOPS está implementado en INTERLISP-D y opera sobre redes de estaciones (puestos de trabajo) Xerox 1100. Ha sido desarrollado en el Xerox Palo Alto Research Center, como sistema de investigación.



M. 1

El M. 1 es un lenguaje diseñado para representaciones basadas en reglas. Sus características principales son un esquema de control de encadenamiento hacia atrás, y una sintaxis de lenguaje muy semejante a la inglesa. El entorno de soporte contiene herramientas de depuración interactivas orientadas a los gráficos, seguimiento (tracing) de la operación del sistema, facilidades para explicar el proceso de razonamiento del sistema, y mecanismos para la consulta automática del usuario, cuando la base no contiene suficiente información. El M. 1 está implementado en PROLOG y opera sobre un ordenador personal IBM, con un sistema operativo PC DOS. Fue desarrollado por Teknowledge como sistema comercial.



OPS5

OPS5 es un lenguaje de ingeniería del conocimiento concebido para representaciones basadas en reglas. Sus principales características son un diseño que soporta generalidad, tanto en la representación de los datos y como en las estructuras de control, una capacidad de comprobación de patrones (pattern-matching) y un intérprete de encadenamiento hacia adelante eficaz, para poder confrontar las reglas con los datos. El entorno de soporte contiene paquetes de depuración y edición, con un mecanismo para ayudar a determinar por qué la regla no se disparó cuando el programador pensaba que lo haría. El OPS5 ha sido implementado en MACLISP y FRANZ LISP. Es uno de los lenguajes de ingeniería del conocimiento más utilizados. Fue desarrollado por la Universidad Carnegie-Mellon, como parte de la familia de lenguajes OPS desarrollados para Inteligencia Artificial y aplicaciones de psicología cognoscitiva. Puede, sin embargo, adquirirse en el mercado, ya que está comercializado por Verac Corporation (OPS5e) y por DEC (OP55).



OPS83

El OPS83 es un lenguaje preparado para representaciones basadas en reglas y orientadas a procedimientos. Su característica principal es la in-

tegración del paradigma de programación basada en reglas con encadenamientos hacia adelante, y el paradigma de programación procedural. El OPS83 esencialmente es un lenguaje de tipo PASCAL ampliado con las construcciones (típicas en elementos y reglas OPS5) de la memoria de trabajo. El OPS 83 también proporciona facilidades para tipos de datos definidos por el usuario, y permite a éste definir procedimientos de resolución de conflictos, regímenes de control, y rutinas de tracing. El OPS83 opera sobre ordenadores DEC VAX 11/750 y 11/780. Fue diseñado en la Universidad Carnegie-Mellon como sistema de investigación, pero puede adquirirse ya que está comercializado por Production Systemas Technologies, Inc.



Personal Consultant

Es un lenguaje de ingeniería del conocimiento concebido para representaciones basadas en reglas, pero también soporta métodos de representación basados en marcos. Sus características principales son: esquemas de control con encadenamiento hacia adelante y hacia atrás, mecanismos de tratamiento de las certezas, jerarquías de clase con herencia, y la posibilidad de acceder a funciones de LISP definidas por el usuario. El Interfaz del usuario incluye un dispositivo orientado a ventanas, que utiliza un sistema de representación en color y facilidades de explicación. El entorno del soporte contiene un editor de base de conocimientos, una facilidad de seguimiento (trace), y también una facilidad de comprobación de regresiones. PERSONAL CONSULTANT está escrito en IQLISP (otro dialecto del LISP) y opera sobre el ordenador TI Professional, el TI Explorer, y otros microordenadores compatibles MS-DOS. Fue desarrollado por Texas Instruments Inc., como sistema comercial.



Prism

PRISM es un lenguaje de ingeniería del conocimento diseñado para representaciones basadas en reglas. Sus características principales son: esquemas de control con encadenamiento hacia adelante y hacia atrás, mecanismos de manejo de certezas y la posibilidad de organizar la base de conocimientos en estructuras jerárquicas, cada una de las cuales va provista de su propia máquina de inferencia y estrategia de control. El entorno de soporte consiste en un programa editor integrado para crear y mantener la base de conocimientos mediante reglas de tipo Inglés. PRISM está implementado en PASCAL y opera sobre ordenadores IBM 370, con sistema operativo VC/CMS. Fue desarrollado por IBM Palo Alto Scientific Center, como sistema de investigación (PRototype Inference SysteM).



Rosie

ROSIE es un lenguaje concebido para representaciones basadas en reglas, pero también puede soportar métodos de representación orientados a procedimientos. Sus principales características incluyen una sintaxis tipo Inglés, una estructura orientada al procedimiento que permite subrutinas anidadas y recursivas, facilidades de concordancia de patrones potentes, y un interfaz con el sistema operativo local que permite a ROSIE controlar tareas remotas. El entorno de soporte va provisto de herramientas de depuración y edición. ROSIE está implementado en INTERLISP y se está convirtiendo para que funcione en el lenguaje de programación C. ROSIE ha sido desarrollado por The Rand Corporation como sistema de investigación. (Rule Oriented System for Implementing Expertise).



Rulemaster

Es una ayuda en la construcción de sistemas, para el desarrollo de sistemas expertos basados en reglas. Está formado por un conjunto de herramientas integradas, que incluyen un lenguaje extensible llamado RADIAL, para expresar las reglas, y un sistema de inducción de reglas que las crea a partir de ejemplos, estructuración jerárquica de las reglas generadas, capacidad de acceder a datos externos y procesos a través de los programas escritos en cualquier lenguaje soportado sobre Unix, generación automática de explicaciones en forma de sentencia completas inglesas, y operadores y tipos de datos definibles por el usuario. RULEMASTER está implementado en el lenguaje de programación C, y opera sobre mini y microordenadores (como las estaciones de trabajo VAX y SUN), que funcionan con el sistema operativo Unix. Fue desarrollado por Radian Corporation como sistema comercial.



Rulewriter

RULEWRITER es una ayuda de construcción de sistemas que asiste a los ingenieros de conocimientos a formular reglas en el lenguaje EXPERT. RULE WRITER utiliza conocimientos sobre ejercicios propuestos en procesos de aprendizaje (casos de entenamiento), una taxonomía de asociaciones posible en el dominio, y mecanismos causales para producir un modelo que clasifica correctamente los casos de entrenamiento sobre la base de las ligazones preestablecidas que mantiene almacenadas. Sus conocimientos causales asociacionales guían un proceso de inducción de reglas, realizado sobre los ejemplos de los casos de entrenamiento. Las reglas de

clasificación aprendidas en los casos de entrenamiento están expresadas directamente en EXPERT. El RULE WRITER está implementado en LISP. Fue desarrollado en la Universidad de Rutgers como sistema de investigación.



Seek

SEEK es un sistema para la construcción de sistemas que le aconseja sobre el refinamiento de las reglas durante el desarrollo de un sistema experto tipo diagnóstico. El sistema para refinar reglas está representado en el lenguaje EXPERT pero está expresado en formato tubular. El SEÉK le sugiere modos posibles para generalizar o especializar reglas (buscando regularidades en sus características) en el cuerpo de los casos almacenados, que tienen conclusiones conocidas. El sistema es interactivo, le sugiere el tipo de cambio (generalización o especialización) y cuáles son los componente que se deben cambiar, pero permite que el usuario decida exactamente cómo generalizar o especializar esos componentes. Este método de refinamiento funciona mejor cuando el conocimiento del experto es muy preciso, y pequeños cambios realizados en la base de conocimientos pueden llevar a mejoras significativas en las características del sistema experto. SEEK está implementado en FORTRAN, y fue diseñado originalmente para operar sobre un sistema DEC-20. Ha sido desarrollado en la Universidad de Rutgers como sistema de investigación.



SRL+

SRL+ es un lenguaje especialmente adaptado para representaciones basadas en marcos. También soporta otros métodos de representación: orientada al objeto y basada en reglas o basado en la lógica. Sus principales características son: relaciones de herencia definibles, conexiones procedurales, un mecanismo de agenda, un lenguaje de simulación discreto y una facilidad de manejo de errores definible por el usuario. El entorno de soporte incluye un sistema de control de la base de datos propio, soporte para producir «gráficos de empresa» en 2-D y un interfaz basado en el analizador de lenguaje natural PLUME. El SRL + está implementado en COMMON LISP, y FRANZ LISP y opera sobre estaciones de trabajo del Carnegie Group, estaciones DEC VAXs bajo VMS y Symbolics 3600. Fue desarrollado por el Carnegie Group Inc. como sistema comercial. (Schema Representation Language +).



S. 1

El S. 1 es un lenguaje de ingeniería del conocimiento diseñado primariamente para representaciones basadas en reglas, aunque también puede soportar métodos de representación orientados a procedimientos y basados en marcos. Sus características principales son un esquema de control de encadenamiento hacia atrás, mecanismo de manejo de certezas y bloques de control, que soportan representaciones orientadas al procedimiento y métodos de programación. El entorno de soporte contiene una facilidad de explicación, y herramientas de depuración orientadas a gráficos, con posibilidades de detención y seguimiento (tracing), durante las consultas. El S. 1 está escrito en INTERLISP y opera sobre estaciones de trabajo Xerox 1100 y 1108. Fue desarrollado por Teknowledge como sistema comercial (System 1.).



Teiresias

TEIRESIAS es una ayuda en la construcción de sistemas que facilita la transferencia interactiva de conocimientos del experto, a la base de conocimientos. El sistema es interactivo con el usuario en un subconjunto restringido del Inglés, para poder ir adquiriendo nuevas reglas sobre el dominio del problema. TEIRESIAS también aporta ayudas en la depuración de la base de conocimientos y, utilizando mecanismos para explicaciones y disponiendo también de un sistema de comprobación de consistencias sencillo. TEIRESIAS está implementado en INTERLISP. Fue desarrollado en la Universidad de Stanford, como sistema de investigación. (Se le ha puesto ese nombre porque Teiresias era ciego en la tragedia griega Edipo Rey.)



Timm

TIMM es una ayuda en la construcción de sistemas que asiste al experto en la construcción de un sistema experto. El experto suministra una lista con todas las decisiones posibles que se pueden tomar, y los nombres y valores de los factores a considerar para llegar a una decisión. En ese momento, el sistema le pedirá ejemplos de factores y sus valores correspondientes que llevan a cada una de las decisiones, utilizando los ejemplos para interferir un conjunto de reglas tipo SI-ENTONCES que permitan obtener las mismas decisiones. TIMM soporta la utilización de ciertos valores para representar la información probabilística. Fue desarrollado en FORTRAN y opera sobre VAX 11/780, y otros ordenadores. TIMM fue de-

sarrollado por General Research Corporation como sistema comercial (The Intelligent Machine Model).



T. 1

El T. 1 es una ayuda en la construcción de sistemas en forma de paquete tutor. Proporciona una introducción a la ingeniería del conocimiento para profesionales técnicos y directivos. Lleva incluidas lecturas en videotape, ejercicios de laboratorio, sistemas de software demostrativos, y material de lectura. Los sistemas de software utilizados para mostrar los conceptos de ingeniería del conocimiento básicos utilizan una versión modificada del lenguaje M. 1 de Tecknowledge. Operan sobre IBM PC con sistema operativo PC DOS 2.0. El T. 1 fue desarrollado por Teknowledge como sistema (o paquete) comercial.

GUIA DE REFERENCIA 5



A continuación incluimos, como complemento de la información facilitada en capítulos anteriores, una guía extensa de referencia de los conceptos utilizados en los sistemas basados en el conocimiento. Se incluyen los términos utilizados en este volumen (para consulta rápida) y otros usados en ingeniería del conocimiento que, mediante una descripción sucinta pero clara, faciliten un complemento para conceptos y temas no tratados aquí.

Acción. En un sistema de producción con arquitectura de encadenamiento hacia adelante se llama acción al lado de la derecha de una regla que está formado por una secuencia de órdenes, cada una de las cuales realiza alguna actividad, como crear, borrar elementos en la memoria de datos, realizando entradas/salidas, modificando la memoria de producción y deteniendo el ciclo del acto de reconocimiento. Cuando una regla se dispara, las acciones que constituyen el lado de la derecha se realizan de forma secuencial, usando las uniones que fueron creadas cuando se instanció la regla.

Activación. El nivel de activación para cada objeto de una red de activación es un número asociado que representa el grado de atención que recibirá tal objeto. Las activaciones se propagan entre objetos relacionados en la red. Véase activación dirigida y activación difundida.

Activación difundida. En una red de activación, la activación difundida es un método de cambio de patrones de activación o atención en la red, tal que la activación fluye hacia fuera de los nodos activos, activando aquello nodos que están conectados directa o indirectamente con él. La activación se produce durante uno o más ciclos de activación sucesivos, y se difunde al siguiente conjunto de nodos que se nos han reactivado hasta el momento.

Activación dirigida. En una red de activación (ver más adelante), la activación dirigida es un método de propagación de la activación de un objeto a otro. La red de activación es un gráfico dirigido que indica que la propagación debe proceder sólo si está de acuerdo con los arcos dirigidos.

Adquisición de conocimiento. Proceso de extracción, estructuración y organización del conocimiento, partiendo de alguna fuente de conocimiento (generalmente una persona experta), y que puede utilizarse en un programa.

Agenda. Lista ordenada de acciones. Mecanismo de control que mantiene una «cola» de prioridades en cuanto a las tareas que deben realizarse. La prioridad de cualquiera de las tareas se puede alterar dinámicamente.

Algoritmo. Procedimiento sistemático y formal que garantiza una solución correcta y óptima de un problema en un tiempo infinito. En un programa convencional, el programador debe especificar el algoritmo que debe seguir el programa. Si se contrasta esta definición con la resolución heurística de problemas, se observa que el término indica un procedimiento claro que garantiza la solución, si existe, o determina que no existe solución.

Algoritmo de concordancia de Rete. Algoritmo para determinar con eficacia cuáles son las reglas que se pueden satisfacer por el contenido de memoria de trabajo, en cada ciclo de reconocimiento, considerando las ligazones entre patrones y datos. Este algoritmo aprovecha la redundancia de los sistemas de producción, almacenando resultados parciales del cómputo de la concordancia, de forma que no necesitan volver a computarse más tarde. Fue desarrollado por Charles L. Forgy y le puso este nombre porque se utiliza en inglés una palabra antigua «Rete» (aunque la palabra usual hoy en día sea «network»).

Antecedente. Lista de las condiciones necesrias para extraer una conclusión. En un sistema de producción, el lado izquierdo de la regla codifica las condiciones antecedentes para que la regla se active, mientras que el lado derecho codifica el consecuente.

Aprendizaje. Cualquier cambio en un sistema que altere sus características a largo plazo. En los sistemas de producción, el aprendizaje produce la adición, eliminación o modificación de las reglas.

Arbol. Gráfico en el que sólo existe un camino entre dos nodos distintos.

Arbol de comprobaciones. Estructura de datos en forma de árbol, en el que el nodo raíz representa el teorema que debe ser probado, y los nodos hijos los teoremas que una vez comprobados bastarán para probar el teorema padre. Los árboles de comprobaciones pueden ser árboles and/or.

Arbol de contexto. También llamado árbol objeto. En EMYCIN, el árbol de contexto constituye la espina dorsal del programa de consulta. Consiste en una organización estructurada de los objetos (contextos) o entidades conceptuales que constituyen el dominio de consulta. Puede existir uno o más contextos. El árbol de contextos estático es una organización de tipos de contextos (por ejemplo, un paciente al que se le han preparado cultivos). Un árbol de contexto dinámico es un conjunto de instancias o especificaciones de contextos (por ejemplo, Juan Pérez con cultivo realizado por la mañana y otro cultivo realizado por la tarde).

Arbol de objetivos. Estructura de datos en forma de árbol, en la que el nodo raíz representa un objetivo que debe alcanzarse, y las ramas hijas de cada objetivo representan subobjetivos, que cuando se alcancen serán suficientes para alcanzar el objetivo representado por su padre. Los árboles de objetivos pueden ser árboles and/or.

Arboles and/or (y/o). El árbol para la prueba de teoremas, y en general el árbol de objetivos, en la resolución de problemas generales son árboles en los que cada nodo lleva su etiqueta indicando si se trata de un nodo and (y) o un nodo or (o). En los nodos and cada uno de los nodos hijos especifica las subpruebas o subobjetivos necesarios que deben llevarse a cabo, si se desea alcanzar el nodo progenitor. En los nodos or, cada uno de los nodos hijos especifica una subprueba o subobjetivo alternativo, y sólo es necesario que se lleve a cabo uno para que se realice el nodo padre.

Arquitectura «de pizarra» (Blackboard). Sistema experto diseñado para representar y controlar los conocimientos. Se basa en unos grupos de reglas independientes, llamadas fuentes del conocimiento, que se comunican a través de una memoria de trabajo o base de datos llamada «pizarra». Un sistema de control basado en la agenda examina todas las acciones pendientes y selecciona la que va a probarse a continuación.

Asociación (-binding). Asociación entre una variable y un valor para esa variable que se mantiene dentro del alcance de una regla, una llamada a la función, o la llamada a un procedimiento. Se llama también «ligadura».

Atomo simbólico. Tipo de dato que permite sólo las operaciones primitivas de asignación y comprobación de igualdades. Se considera que dos

átomos simbólicos son iguales si tienen el mismo nombre impreso (la misma secuencia de caracteres alfabéticos y especiales, que se utiliza para especificar la identidad del átomo).

Atributo. Ver Propiedad.

Atributo de vector. En OPS5, atributo que puede tomar una secuencia de valores atómicos.

Atributo multivalente. Atributo que puede tener más de un valor. Si, por ejemplo, un sistema busca valores para el atributo restaurante, y éste es multivalor, se podrán identificar dos o más restaurantes.

BNF. Siglas de Backus-Naur Form: (ver Forma Backus-Naur).

Back chaining. Véase encadenamiento hacia atrás.

Backtracking dirigido a dependencias. Técnica de programación que permite a un sistema eliminar los efectos de suposiciones incorrectas durante la búsqueda de solución a un problema. Cuando el sistema infiere nueva información, mantiene los registros de dependencia de todas sus deducciones y suposiciones, mostrando cómo fueron derivadas. En el momento en que se encuentra que una suposición fue incorrecta, realiza un backtraking (salto hacia atrás) en las cadenas de inferencias, eliminando aquellas conclusiones basadas en la suposición incorrecta.

Backtracking. Técnica de búsqueda que va seleccionando en cada nodo del gráfico el nodo adyacente que se tomará a continuación. Si éste es un nodo que no satisface las condiciones que se están comprobando —«fallo»—, se tomará otro nodo adyacente al original. Si todos los nodos adyacentes son «fallos», el propio nodo considerado será «fallo». De esta forma, a través de la secuencia de inferencias se van probando los distintos caminos. En la planificación de problemas, este sistema se utiliza para ir probando uno tras otro todos los planes, de forma que vayan identificándose los que sean inaceptables.

Base de conocimientos. Una parte del sistema basado en el conocimiento, o sistema experto que contiene el conocimiento sobre el dominio de que se trate.

Base de conocimientos dinámica. Véase Memoria de datos.

Base de datos. Conjunto de hechos, aseveraciones y conclusiones que se utilizan para concordar las partes condicionales (IF) de las reglas, en un sistema basado en reglas.

Blackboard. Véase pizarra.

Bloque (Chunk). Conjunto de hechos que se almacena y recupera como una unidad. Para indicar las limitaciones de la memoria de trabajo, se suele utilizar como indicador el número de bloques que se pueden manejar simultáneamente.

Búsqueda en anchura. En una jerarquía de reglas u objetos, este tipo de búsqueda indica una estrategia en la que todas las reglas u objetos dentro del mismo nivel de jerarquía se examinan antes que cualquiera de las reglas u objetos del siguiente nivel inferior. En contraposición a búsqueda en profundidad.

Búsqueda en profundidad. En un sistema jerárquico de reglas u objetos, la búsqueda en profundidad indica una estrategia en la cual se examina una regla u objeto del nivel más alto y a continuación las reglas u objetos situados inmediatamente debajo del examinado. Al proceder de este modo, el sistema busca por una rama simple del árbol jerárquico, hasta terminarla. En contraposición a búsqueda en anchura.

Búsqueda exhaustiva. Dícese de una búsqueda que es exhaustiva si se examina cada una de las ramas del árbol de decisión o red de decisiones. (Este tipo de búsqueda es prácticamente imposible o muy costosa, por multitud de razones, en algunos casos concretos). Los sistemas basados en el conocimiento suelen realizar búsquedas exhaustivas en sus bases de conocimiento.

C. Lenguaje de programación, de propósito general, de bajo nivel, y eficaz asociado con el sistema operativo UNIX. Normalmente se utiliza para programación de sistemas.

CAD. Computer Aided-Design-Diseño Asistido por Ordenador. Tecnología informática creada para asistir al diseñador durante el proceso de diseño; por ejemplo, de circuitos integrados, de piezas, etc.

CAI. Computer Aided Instruction. Aplicación de los ordenadores en la Educación. El ordenador controla y sigue el aprendizaje del alumno, que va contestando a las preguntas propuestas, o resolviendo situaciones, o jugando a los juegos propuestos. El sistema va ramificando según sean las respuestas del alumno. Estos programas no son inteligentes, en el sentido de que no desarrollan un modelo de cómo conoce el alumno la materia, y tampoco puede individualizarse a cada alumno, normalmente.

Cadena de inferencia. Secuencia de pasos o reglas de aplicación utilizados por un sistema basado en reglas para llegar a conclusiones.

Cálculo de predicados. Lenguaje formal de la lógica clásica que utiliza funciones y predicados para describir las relaciones entre entidades individuales.

Certeza. Grado de confianza en un hecho determinado o en una relación. Se representa por un número, dentro de una gama de valores, en el que el superior denota una certeza absoluta de que la aseveración es cierta, y el inferior la certeza también absoluta de que la aseveración es falsa. En IA, este término se utiliza en contraposición a probabilidad, que indica las posibilidades existentes de que ocurra un determinado acontecimiento. Este término es sinónimo de Factor de confianza.

Ciclo. Cada paso de iteración de un bucle. En sistemas de producción, una ejecución está formada por ciclos de actos de reconocimiento reiterados. En el algoritmo de concordancia de Rete, los ciclos de concordancia tienen lugar cada vez que se añade o se elimina un elemento de la memoria de trabajo. En sistemas de activación difundida, la activación se propaga de forma incremental durante las series de ciclos de activación.

Ciclo de activación. En una red de activación, el ciclo de activación es el período de tiempo durante el cual se propaga la activación entre los objetos adyacentes (contiguos). Normalmente, durante cada uno de los ciclos de activación se propaga un arco a partir de la fuente de activación. Durante un único ciclo de reconocimiento, se llevan a cabo uno o más ciclos de activación.

Ciclo de concordancia. Etapa de un proceso que tiene lugar en el algoritmo de concordancia de Rete, siempre que se produce un cambio en la memoria de trabajo. Como resultado, se produce una actualización de la red de Rete, y del conjunto de conflictos.

Ciclo de reconocimiento. Bucle interactivo que tiene tres fases: concordancia, resolución de conflictos y disparo de las reglas.

Clase de elementos. Tipo de datos de un elemento de la memoria de trabajo.

Cláusula de Horn. En programación lógica, las cláusulas de Horn son expresiones unidas por «or» con, al menos, una proposición positiva. Así, una cláusula de Horn tiene la forma siguiente «No A o No B o... o no C o D». La programación lógica se realiza con una mayor eficacia restringiendo el tipo de aserciones lógicas a cláusulas de Horn, del mismo modo que los sistemas de producción exigen normalmente disponer del conocimiento en forma de reglas si-entonces.

Common LISP. Dialecto del LISP que supone una versión estandarizada ejecutable en muchas máquinas. Los primeros esfuerzos realizados tuvieron muchas dificultades, ya que el LISP es un lenguaje muy adaptable, por lo que la mayor parte de los usuarios se resisten a adaptarlo de forma específica para sus máquinas.

Compilador. Programa que traduce un programa (fuente escrito en un lenguaje de alto nivel) en programa objeto, en lenguaje de bajo nivel. Si el programa ha sido compilado, normalmente se ejecutará más rápidamente que si el mismo programa es interpretado.

Composición. Mecanismo de aprendizaje que combina dos o más reglas que actúan en secuencia para producir una regla única cuyo efecto es la resultante de las reglas componentes. El nombre deriva por analogía del concepto matemático de composición de funciones.

Concordancia. En un sistema de producción, el proceso de concordancia compara un conjunto de patrones que aparecen en el lado izquierdo de las reglas, con los datos de la memoria de datos, para ver todos los caminos posibles en los que se pueden satisfacer las reglas con ligaduras consistentes.

Concordancia parcial. Conjunto de asociaciones entre los elementos de condición (antecedentes de las reglas) y los elementos de la memoria de trabajo, que satisfacen parcialmente el lado izquierdo de la regla. No todos los elementos de condición necesitan ser contrastados. En algunos casos, se da un umbral para especificar el número mínimo que debe ser contrastado.

Condición. a) Antecedente de una regla. b) Elemento de una condición. c) Proposición que resume el estado de ejecución de un programa.

Conjunto conflicto. Conjunto de todas las instanciaciones por el proceso de concordancia durante un ciclo de reconocimiento. El proceso de resolución de conflictos selecciona una instanciación del conjunto de conflictos y la activa para su solución.

Conocimiento. Conjunto de hechos y relaciones que, adecuadamente estructurados en forma de reglas, representan el bagage de informaciones que un experto suele tener sobre una tema determinado.

Conocimiento compilado. Conocimiento que codifica las reglas de inferencia en las que se han eliminado las cadenas de rozamiento implícitas, para lograr una mayor eficacia.

Conocimimiento declarativo. Conocimiento que puede ser ejecutado inmediatamente. Para ser efectivo debe ser interpretado por el conocimiento procedural.

Conocimiento del dominio. Conocimiento específico de un dominio concreto.

Conocimiento experimental. Conocimiento obtenido a través de la experiencia. Generalmente consiste en hechos y reglas elementales (conocimientos superficiales). En contraposición con conocimientos profundo o teorías o principios formales.

Conocimiento profundo. Conocimiento de los principios y teorías básicas, axiomas y hechos relacionados con un dominio. En contraposición con conocimiento superficial.

Conocimiento superficial. Conocimiento heurístico o experimental. Conocimiento adquirido mediante la experiencia que se utiliza en la resolución de problemas prácticos. Se refiere generalmente a hechos específicos y teorías sobre algún dominio particular, o tarea, pero principalmente está compuesto por reglas elementales.

Conocimientos de metanivel. Es el conocimiento que tiene un sistema de ordenador sobre sí mismo, la extensión y fiabilidad de sus informaciones, y cuándo y cómo utiliza mejor sus conocimientos sobre el dominio.

Consecuencia. Conclusión de una regla o proposición lógica. En un sistema de producción, el lado izquierdo de la regla codifica las condiciones antecedentes de activación de dicha regla, y el lado derecho codifica la consecuencia.

Contexto. Un estado, dentro del proceso de resolución del problema. En un sistema de producción, se puede representar el conjunto por un elemento de la memoria de trabajo específico, al que se designa con frecuencia con el nombre de elemento del contexto o elemento de control o subobjetivo. Es frecuente que las tareas aislables que deban realizar los sistemas de producción puedan partirse en subtareas que una vez iniciadas puedan ejecutarse hasta completarse. Las reglas que constituyen esta tarea tienen cada una de ellas elementos de condición que deben concordar con el elemento de contexto asociado.

Control (de un sistema basado en el conocimiento). Es el método utilizado por la máquina de inferencias para regular el orden en el que se pro-

duce el razonamiento. El encadenamiento hacia atrás, hacia adelante, y las búsquedas en profundidad o anchura son ejemplos de métodos de control.

Data driven. Expresión inglesa para designar el encadenamiento hacia adelante o encadenamiento guiado por los datos.

De abajo a arriba (bottom up). Estrategia de actuación, que va del análisis de lo sencillo y concreto a lo complejo y abstracto. Si se aplica a la estrategia de resolución de problemas, se refiere al método de comenzar a acumular resultados que proceden de observaciones sencillas o de hechos, e ir procediendo con combinaciones más complejas o hipótesis. Los sistemas de producción con arquitectura de encadenamiento hacia adelante suelen utilizar este tipo de estrategia de resolución de problemas. Si se aplica en la metodología de programación, este término se refiere a un estilo de programación, en la que los componentes simples del programa se escriben antes que los complejos. La estrategia opuesta es de arriba a abajo.

De arriba a abajo (top down). Estrategia de procedimiento por la cual se va de lo complejo y abstracto a lo sencillo y concreto. Si se aplica estrategia en la resolución de problemas, se refiere al método de comenzar con un problema complejo e ir descomponiéndolo en subproblemas que son más sencillos de resolver. Los sistemas con encadenamiento hacia atrás suelen resolver los problemas por este procedimiento. Si se aplica a la metodología de programación, el término se refiere a un estilo en el cual la organización abstracta del programa se va determinando en etapas en las que va aumentando la concreción, culminando por escribir los elementos primitivos que se pueden ejecutar directamente. La estrategia o puesta es de abajo a arriba.

Definido por el usuario. Unidades de programa que amplían un lenguaje de implementación. Muchos lenguajes de programación permiten la definición de estructuras de programa complejas que pueden referenciarse como una unidad. Los lenguajes como el PASCAL, C y OPS 83 permiten al programador definir el nombre y el acceso a tipos de datos no primitivos. En casi todos los segmentos de programa de los lenguajes (procedimientos, funciones, subrutinas) se pueden definir, nombrar e invocar si se utiliza una sintaxis idéntica a la usada en la construcción de las funciones. A estas unidades, no primitivas, se les designa como tipos definidos por el usuario, y funciones definidas por el usuario.

Demonio. Procedimiento que se ejecuta siempre que un predicado específico sobre la base de datos sea verdadero. En un sistema de produc-

ción, una regla que no está restringida a dispararse dentro de cualquier contexto especial, o como parte de un motivo. Una regla demonio puede dispararse nada más ser instanciada, sin considerar la presencia de ningún objetivo parcialmente completo. El nombre viene de la unidad de un programa de percepción llamado Pandemonium.

Dependencia. Relación entre una consecuencia y sus antecedentes, que debe ser almacenada si el proceso de razonamiento tiene que examinarse retrospectivamente. Se llama red de dependencias a un gráfico en el cual los nodos representan aserciones y los arcos u otros representan relaciones.

Dirigido a un modelo. Otra forma de designar el encadenamiento hacia atrás. En contraposición a controlado o dirigido por los datos.

Dirigido hacia objetivos. Término que significa lo mismo que el Encadenamiento hacia atrás. Se utiliza en contraposición a dirigido a datos.

Dirigido por datos. Controlado por los cambios efectuados en los datos, en lugar de hacerlo por los cambios de objetivo. En contraposición a dirigido por objetivos. Véase demonio y encadenamiento hacia adelante.

Dirigido o controlado por un patrón. Controlado por ciertas configuraciones de datos. Los sistemas de producción son un caso especial de sistemas controlados por patrones.

Discriminación. Proceso de aprendizaje que distingue las instancias de un concepto de las no instancias. Se llama también discriminación al acto de refinar una supergeneralización de un concepto que debe aprenderse para excluir las no instancias que fueron clasificadas erróneamente como instancias del concepto. También puede referirse a un mecanismo de aprendizaje específico que explota la retroalimentación relacionada con las clasificaciones erróneas, para refinar una supergeneralización de un concepto.

Disparar. Ejecutar el conjunto de acciones especificadas en el lado derecho de una regla o instanciación. Este término deriva de la neurofisiología, ya que se dice que las neuronas se «disparan» cuando generan una acción.

Distintivo. Instancia de un tipo. También puede ser un átomo único que puede utilizarse como etiqueta, o bien un símbolo utilizado en el al-

goritmo de concordancia de Rete para representar un elemento de la memoria de trabajo.

Dominar. Se dice que la primera instanciación domina a la segunda, cuando una estrategia de resolución de conflictos deja una instanciación en el conjunto conflicto, pero elimina la segunda instanciación.

Dominio. En matemáticas, conjunto de valores que debe asumir el argumento de una función. En sistemas expertos, el campo del conocimiento o clase de las tareas que puede abordar.

Dominio de tarea. En sistemas expertos, otra forma de designar el término dominio.

EMYCIN. La primera herramienta de construcción de sistemas expertos. EMYCIN deriva del sistema experto MYCIN. Después de desarrollar MYCIN, los diseñadores consideraron que podían extraer el conocimiento médico específico de MYCIN (es decir el CYCIN esencial). El módulo-concha (shell) resultante consiste en una máquina de inferencias de encadenamiento hacia atrás, un director de consultas y varias ayudas de adquisición de conocimiento. Esta «concha», o herramienta, podía combinarse con otra base de conocimientos para crear un nuevo sistema experto.

Ejecutar. Llevar a cabo las fases especificadas en el procedimiento, o las acciones de una regla.

Elemento. Es la unidad más primitiva en la que puede descomponerse un sistema. En OPS5, por ejemplo, las unidades de los lados izquierdos de las reglas se llaman elementos de condición, y las unidades de la memoria de trabajo se llaman elementos de la memoria de trabajo.

Elemento atributo-valor (A-V). Estructura de datos para los elementos de la memoria de datos, que codifican el conocimiento sobre los objetos en forma de un conjunto de pares ordenados, de los cuales el primer elemento especifica la identidad de un atributo, y el segundo el valor que toma ese atributo para ese objeto.

Elemento de contexto. Elemento de la memoria de trabajo que indica el estado de cómputo (contexto) y que se utiliza como control. Este elemento es una instancia de un elemento de control.

Elemento de control. Elemento de la memoria de trabajo cuya única

utilidad es almacenar conocimientos de control. Como ejemplo de elemento de control, citaremos un elemento de contexto.

Elemento de la memoria de trabajo. Es la unidad de la memoria de trabajo. En los lenguajes orientados al objeto, los elementos de la memoria de trabajo son elementos de valor-atributo.

Elemento de una condición. En algunos casos, el lado izquierdo de una regla en un sistema de producción se expresa como conjunto de patrones (o máscaras) que deben contrastarse con el contenido de la memoria de datos. Se denomina elemento de una condición a cada uno de estos patrones. Cuando se instancia una regla, cada elemento de condición debe ser tal que concuerde con un elemento de la memoria de datos.

Elemento resultante. Estructura temporal utilizada en los lenguajes orientados al objeto para construir una descripción de un elemento de la memoria de trabajo con la finalidad de modificar la memoria de trabajo. Las acciones de llamada y modificación se implementan utilizando elementos resultantes.

Eliminación de orden temporal. Ver Recency.

Encadenamiento hacia adelante. Método de resolución de problemas, que comienza con un conocimiento inicial y aplica reglas de inferencia para generar nuevos conocimientos hasta que cualquiera de las inferencias satisface el objetivo, o no se pueden realizar más inferencias. En los sistemas de producción de encadenamiento hacia adelante, la aplicabilidad de una regla viene determinada por la satisfacción de las condiciones especificadas en su lado izquierdo, respecto del conocimiento almacenado actualmente en la memoria de datos.

Encadenamiento hacia atrás. Método de resolución de problemas. Es una de las distintas estrategias de control que regulan el orden en el que se establecen las inferencias. El sistema comienza marcando un objetivo a cumplir, y se va ampliando cada objetivo sin resolver en otros subobjetivos hasta encontrar la solución, o hasta que cada objetivo ha sido dividido en sus componentes más sencillos. Cuando se resuelve un subobjetivo, se pasa al objetivo padre. En los sistemas de producción de arquitectura de encadenamiento hacia atrás, la aplicación de una regla se determina examinando sus conclusiones (en el lado derecho) en lugar de sus condiciones antecedentes (en el lado izquierdo).

Entrada/Salida. Comunicación entre el programa de ordenador y el usuario de dicho programa.

Escalar. En los lenguajes de programación, uno de los tipos de datos primitivos, generalmente, un entero, un número en coma flotante, un carácter o un valor lógico o átomo simbólico. No puede ser una lista, una matriz (vectorial) o un registro (estructura).

Espacio de búsqueda. Equivalente a espacio de un problema.

Espacio de un problema. Gráfico en el que los nodos representan todos los estados posibles de las soluciones parciales o completas del problema, y los arcos representan los operadores que transforman un estado en otro. La búsqueda de la solución al problema que se está considerando se representa estableciendo un camino a partir del nodo que representa el estado inicial, hasta el nodo que representa el estado objetivo.

Especificidad. Estrategia de resolución de conflictos que prefiere instanciaciones de reglas más específicas, medidas por lo general por el número de variables o constantes o números de las comprobaciones del lado izquierdo. Este principio incluye el heurístico de que las reglas con antecedentes más detallados son más discriminatorias que las que lo son menos, produciendo por lo general mejores resultados.

Esquema. Formalismo para representar una información sobre un concepto único, utilizando propiedades relacionadas con dicho concepto. Las propiedades se suelen representar mediante una ranura, y pueden consistir en procedimientos relacionados con ellas para computar las propiedades que no están disponibles inmediatamente.

Estabilidad. Continuidad en el comportamiento de un sistema. La estabilidad de un sistema de producción se ve influenciada por su estrategia de resolución de conflictos.

Estación o puesto de trabajo. Término inglés que se refiere a sistemas de ordenador que pueden ayudar al usuario a realizar su trabajo. Este trabajo puede ser, en principio, cualquiera.

Estrategia de control. Método para seleccionar la acción siguiente entre varias fases alternativas de resolución de un problema. En sistemas de producción, el encadenamiento hacia atrás, es un ejemplo de estrategia de control.

Estrategia de resolución de conflictos. Es un principio específico aplicable para poder ordenar parcialmente las instanciaciones del conjun-

to conflicto. Aquellas instanciaciones que aparecen dominadas por otras, de acuerdo con este principio, se eliminan del conjunto de conflictos, y se descarta la posibilidad de activarlas en ese ciclo.

Estructura. Otra acepción de Esquema. Método de representación del conocimiento que asocia las «características» con los nodos que representan conceptos u objetos en una red semántica. Las «características» están descritas en términos de atributos (llamados ranuras) y de sus valores. Los nodos forman una red, conectada por relaciones, y organizada dentro de una jerarquía. Cada ranura puede rellenarse con valores, para así ayudar a describir el concepto que ese nodo representa. El proceso de añadir o eliminar valores de las ranuras puede activar procedimientos (partes de código contenidas en sí mismas) relacionados con las ranuras. Estos procedimientos pueden así modificar los valores de otras ranuras, continuando el proceso hasta alcanzar el objetivo deseado.

Estructura de lista. Estructura formada por una lista. Conjunto de elementos colocados entre paréntesis, en el cual cada elemento puede ser un símbolo, u otra lista.

Experiencia. Eficiencia en un dominio especializado. El sistema experto se dice que tiene experiencia en ese dominio si sus cualidades pueden compararse con las de una persona con un tiempo de cinco a diez años de experiencia en ese dominio.

Experto en un dominio. Persona muy entrenada y eficaz en la realización de las tareas para las que ha construido el sistema experto, y que sirve para articular este conocimiento para beneficio del ingeniero del conocimiento que incorpora el conocimiento del dominio del experto al sistema experto.

Explicación. Proceso por el cual se describe cómo un sistema experto alcanza sus conclusiones o por qué hace una determinada pregunta al usuario. Las explicaciones se pueden utilizar para justificar las decisiones o las estrategias de resolución de los problemas, o también para enseñar dichas estrategias al usuario.

Factor de confianza. Sinónimo de certeza. Véase certeza.

Filtrado. Exclusión de datos (filtración de datos) o reglas (filtración de reglas) del proceso de concordancia, con objeto de mejorar la eficacia.

Filtro de los datos. Restricción de una parte de la memoria de datos

que participa en el proceso de concordancia a un subconjunto, con la finalidad de obtener uña mayor eficacia.

Filtro de reglas. Restricción de la parte de la memoria de producción que participa en el proceso de concordancia con un subconjunto, en aras de una mayor eficacia.

Forma Backus-Naur. Lenguaje formal para expresar gramáticas de contexto libre. Una gramática es un conjunto de reglas de sobreescritura, cada una de las cuales dispone de un lado izquierdo y un lado derecho, separados por el símbolo de metalenguaje ::=. El lado izquierdo de cada regla es un símbolo no terminal de la gramática, y el lado derecho de la secuencia de símbolos no terminales y terminales. Los símbolos no terminales suelen aparecer entre los símbolos <>. Algunas versiones ampliadas de la Forma Backus-Naur incluyen símbolos de metalenguaje adicionales para indicar la repetición y alternación. Este lenguaje lleva su nombre debido a John Backus y Peter Naur, que fueron los introductores de esta forma en la descripción general del lenguaje de programación ALGOL-60.

Fuentes de conocimiento concurrentes. En un sistema experto, módulos especializados que analizan los datos independientemente, y se comunican a través de una base de datos central, llamada generalmente «pizarra».

Generalización. Principio abstracto que captura lo común dentro de un conjunto de instancias específicas. También puede referirse al proceso de derivar un principio abstracto por procedimientos deductivos o inductivos. Puede ser también el mecanismo de aprendizaje por el cual se puede derivar un principio abstracto haciendo a un principio específico más abstracto. Como última acepción, en los programas de aprendizaje implementados como sistemas de producción, un mecanismo de aprendizaje que amplía las condiciones de una regla, de forma que ésta pueda aplicarse a una gama de datos más amplia.

Generar y probar. Técnica de resolución de problemas que utiliza un generador que produce posibles soluciones, y un evaluador.

Gramática de contexto libre. Gramática para describir un lenguaje de contexto libre. Como metalenguaje para expresar una gramática de contexto libre podemos citar la forma de Backus Naur.

Gramática generativa. Gramática formal de un lenguaje, expresado como conjunto de reglas que pueden aplicarse para generar todas las sen-

tencias de dicho lenguaje, y ninguna sentencia que no esté en él. La gramática generativa no es necesariamente el mejor método de representar la gramática si el propósito que se tiene es clasificar las sentencias del lenguaje.

Granularidad. Nivel de detalle de un trozo de información, por ejemplo una regla o una estructura.

Hecho. Instrucción cuya validez es aceptada. En la mayoría de los sistemas del conocimiento, un hecho consiste en un atributo y un valor específico asociado.

Herramienta de construcción de un sistema experto. Lenguaje de programación y paquete de soporte utilizado para construir un sistema experto.

Herramientas de construcción de grandes sistemas híbridos. Una de las clases de las herramientas de la ingeniería del conocimiento que pone el mayor interés en la flexibilidad. Los sistemas se diseñan para construir grandes bases de conocimientos. Generalmente incluyen una colección híbrida de estrategias de control e inferencia distintas. La mayoría de las herramientas híbridas comerciales incorporan marcos y facilitan la programación orientada al objeto.

Herramientas de construcción de grandes sistemas específicos. Es una de las clases de herramientas de ingeniería del conocimiento que sacrifica la flexibilidad para facilitar el desarrollo eficaz de sistemas expertos definidos de una forma mucho más específica. Actualmente la mayoría de las herramientas específicas ponen énfasis en las reglas de producción.

Herramientas de los sistemas del conocimiento pequeños. Estas herramientas se pueden utilizar en ordenadores personales.

Heurístico (regla heurística). Principio o regla elemental que comporta algún conocimiento de resolución de problemas, y que tiene la particularidad de llevar a la solución del problema más rápidamente que el método general de resolución implícito en un determinado algoritmo, reduciendo considerablemente la búsqueda. Sin embargo, no garantizan su funcionamiento en todas las situaciones. Este término fue popularizado principalmente por el matemático G. Polya.

I/O. Véase Entrada/Salida.

IA. Inteligencia Artificial.

INTERLISP. Dialecto del LISP. Entorno de programación que proporciona al programador muchas ayudas para facilitar el desarrollo y mantenimiento de programas de LISP extensos.

Implementación (entorno de implementación). Este concepto se refiere al entorno en el que funcionará el sistema experto. El entorno de implementación incluye el hardware sobre el que funcionará el sistema, el sistema operativo que soportará el sistema experto, los lenguajes de alto nivel de los que dependerá éste, y cualquier interface entre el ordenador y otros sistemas de ordenadores o sensores. Por ejemplo, un pequeño sistema de conocimientos puede ejecutarse en un ordenador personal de IBM con una RAM de 256 K. El sistema utilizará un sistema operativo MS-DOS, y necesitará tener instalado un programa PC LISP PC antes de que pueda pasar a funcionar el sistema de conocimiento. Hasta el momento, la mayor parte de los sistemas expertos y herramientas han sido desarrollados para que funcionen en entornos de implementación muy específicos, y es importante considerar con mucho cuidado los requerimientos antes de decidir si puede utilizar ese sistema específico.

Incertidumbre. En el contexto de los sistemas expertos, este término se refiere a un valor que no puede determinarse durante la consulta. La mayoría de los sistemas expertos pueden manejar las incertidumbres; es decir, permiten al usuario indicar si conoce o no la respuesta. Si se trata de una respuesta dudosa en alguna medida (o que comporta incertidumbre), el sistema utiliza otras reglas para intentar establecer el valor por otros medios, o toma el valor de omisión adecuado.

Indicador de tiempo. Número unido a un elemento de la memoria de trabajo que varía de forma monotónica con el tiempo, y se utiliza para indexar el orden de aparición del elemento, cuando se aplica el algoritmo de eliminación por orden temporal (recency).

Inferencias. Derivación de una proposición a partir de otras proposiciones. Se puede organizar una serie más o menos compleja de inferencias para llegar desde unas proposiciones antecedentes dadas hasta unas consecuentes, mediante un proceso denominado de encadenamiento hacia adelante, o de inferencia dirigida por los datos: o bien, se puede comenzar a partir de la especificación de las conclusiones que se desean obtener, y retroceder intentando probar los antecedentes que las justifiquen, según un proceso que se llama de «encadenamiento hacia atrás», o de in-

ferencia dirigida por un modelo, o también inferencia controlada por los objetivos.

Inferencias controladas por los datos. Inferencias controladas por hechos, en lugar de serlo por objetivos. Véase Encadenamiento hacia adelante.

Inferencias controladas por objetivos. Inferencias controladas por objetivos, en lugar de serlo por los datos. Véase Encadenamiento hacia atrás.

Ingeniería del conocimiento. Proceso de construcción de sistemas expertos.

Ingeniero de Software. Persona que diseña software. Su tarea es similar a la de un ingeniero del conocimiento en el desarrollo de los programas de software convencionales.

Ingeniero del conocimiento. Aquella persona que diseña y construye el sistema experto. Suele ser un experto en la aplicación de métodos de inteligencia artificial.

Instanciación. Patrón o fórmula en la que se han sustituido las variables por constantes. En un sistema de producción, la instanciación es el resultado de confrontar una regla con el contenido de la memoria de datos. Puede presentarse en forma de par ordenado, en el que el primer miembro identifica la regla que se ha satisfecho, y el segundo es una lista de elementos de la memoria de trabajo que satisfacen los elementos condicionales de la regla.

Inteligencia artificial. Subcampo de la Informática que se ocupa del desarrollo de conceptos y métodos de inferencia simbólica mediante ordenador, y de la representación simbólica del conocimiento, para utilizarla en inferencias. En otras palabras, se ocupa del desarrollo de programas que resuelven problemas, «aprenden» experiencias, «comprenden» lenguajes, interpretan escenas visuales, y, en general, se comportan de la forma que los humanos consideran «inteligente».

Interface de usuario. Véase Interface.

Interface. Ligazón entre los programas de ordenador y el mundo exterior. Un programa dado puede disponer de varios interfaces. Los siste-

mas basados en el conocimiento disponen de interfaces para su desarrollo (interface de adquisición de conocimiento) y para los usuarios (interface de usuario). Además, algunos sistemas disponen de interfaces que pasan información hacia o desde otros programas, bases de datos, dispositivos de presentación, o sensores.

Interpretación de imágenes. Utilización de métodos de Inteligencia Artificial para procesar e interpretar imágenes visuales, por ejemplo analizando las señales producidas por una cámara de TV, para conocer y clasificar los tipos de objetos que aparecen en la imagen.

Intérprete. En un sistema experto, aquella parte de la máquina de inferencias que decide cómo aplicar el conocimiento del dominio. En un sistema de programación, aquella parte del sistema que analiza el código para decidir qué acciones deberán llevarse a cabo a continuación.

Intérprete de reglas. En los sistemas de producción, otra denominación de intérprete.

Jerarquía. Conjunto ordenado de conceptos y objetos, en el que algunos están subordinados a otros. Las jerarquías normalmente implican herencias, y así los objetos o conceptos que se encuentran más arriba en la organización, «contienen» los objetos o conceptos que se encuentran debajo.

Jerarquía de herencia. Estructura dentro de una red o sistema de estructura semánticos, que permite a los elementos inferiores en la red heredar propiedades de los elementos más altos en dicha red.

KE. Siglas de Knowledge engineer-Ingeniero del conocimiento.

LISP. Lenguaje de programación basado en el proceso de listas. LISP es el lenguaje seleccionado más comúnmente por los investigadores en Inteligencia Artificial de América.

LSI. Siglas del término inglés Large Scale Integration-Integración a gran escala. Se refiere a los circuitos físicos integrados.

Lado izquierdo. Una de las dos partes de una regla. La otra es el lado derecho. El lado izquierdo especifica los antecedentes que deben satisfacerse si se aplica la regla. En las reglas de gramática, el lado izquierdo especifica una cadena de símbolos que pueden sustituirse por otra cadena

de símbolos. En sistemas de producción que utilizan estrategias de encadenamiento hacia atrás, el lado izquierdo especifica los subobjetivos del objetivo que está especificado en el lado derecho. En los sistemas de producción de encadenamiento hacia adelante, el lado izquierdo consiste en un conjunto de elementos de condición que deben contrastarse con el contenido de la memoria de datos.

Lenguaje de alto nivel. Los lenguajes de ordenador están en una gama que va desde las instrucciones máquina a lenguajes intermedios, como por ejemplo el FORTRAN o COBOL, y finalmente, a lenguajes de alto nivel, como por ejemplo el Ada o el C. Los lenguajes de alto nivel, incorporan construcciones más complejas que los lenguajes más sencillos.

Lenguaje de contexto libre. Lenguaje formal en el que cada sentencia se puede generar mediante una gramática en la que el lado izquierdo de cada regla de reescritura consiste en un único símbolo no terminal.

Lenguaje de ingeniería del conocimiento de propósito general. Lenguaje de ordenador diseñado para construir sistemas expertos e incorporarles características que los hacen aplicables a distintos problemas, áreas y tipos.

Lenguaje natural. Método normal de intercambio de información entre las personas, como puede ser el español, el francés, chino, etc. (en contraposición a los lenguajes artificiales, o lenguajes de ordenador).

Lenguaje orientado a los sistemas de producción. Lenguaje de ordenador que emplea como componente fundamental una arquitectura que es un sistema de producción.

Lenguaje orientado al problema. Lenguaje de ordenador diseñado para una clase de problemas específicos, por ejemplo el FORTRAN diseñado especialmente para realizar cálculos científicos (especialmente, algebraicos), y el COBOL, con características muy útiles para el desarrollo de aplicaciones empresariales.

Ligadura. Dícese de la asociación que se establece entre una variable y un valor, cuando dicha asociación implica el establecimiento de una gama de valores para la variable.

Ligaduras parciales. Conjunto de elementos de la memoria de trabajo y ligaduras que constituyen una ligadura consistente para una secuencia prefijada de elementos de condición de una regla.

Líneas múltiples de razonamiento. En resolución de problemas, técnica en la que se desarrollan en paralelo varios sistemas independientes de resolución del problema.

Lista. Secuencia de objetos definidos recursivamente. En LISP, es una secuencia de cero o más átomos y listas rodeada de paréntesis.

Lógica. Es un sistema que prescribe reglas para manipular símbolos. Muchos sistemas lógicos tienen potencia suficiente para manipular estructuras del conocimiento que incluyen cálculo proposicional y cálculo de predicados.

Lógica difusa. Modelo desarrollado para realizar razonamientos aproximados en el cual los valores y los cuantificadores se definen como distribuciones de probabilidades, con etiquetas linguísticas, como por ejemplo verdadero, muy verdadero, no muy verdadero, bastantes, pocos, y varios. Las reglas de inferencia son aproximadas en lugar de ser exactas, para mejor manipular la información que es incompleta, imprecisa o inconcreta.

MYCIN. Sistema experto desarrollado en la Universidad de Stanford, a mediados de la década de los 70. El sistema fue diseñado para asistir a los médicos en el diagnóstico y tratamiento de la meningitis y otras enfermedades bacterianas. Suele considerarse el primer sistema experto. Existieron otros sistemas que utilizaron muchas de las técnicas asociadas con los sistemas expertos, pero el MYCIN, fue el primero que combinó las características fundamentales, con una clara separación entre la base de conocimientos, y el motor de inferencias. Esta separación, por otro lado, llevó al desarrollo de la primera herramienta de construcción de sistemas expertos: el EMYCIN.

Mantenimiento de la consistencia del conocimiento. Proceso de control de las dependencias entre las aserciones en una base de conocimiento, de forma que pueda servir de ayuda en las deducciones posteriores que se realicen, al ir adquiriendo nueva información.

Mantenimiento de un sistema experto. A diferencia del software de ordenador tradicional, que se actualiza con no demasiada frecuencia, los sistemas expertos, por su propia naturaleza son muy sencillos de modificar. La mayoría de los sistemas expertos que están funcionando se están mejorando constantemente, con la adición de nuevas reglas. En la mayoría de las aplicaciones, la organización del usuario suele desear establecer

una rutina que capture e incorpore nuevos conocimientos al sistema. Puede ser muy útil responsabilizar a una persona para que introduzca nuevas reglas, siempre que los datos o procedimientos cambien, o siempre que aparezcan cuestiones que el sistema vigente no pueda resolver.

Memoria. Uno de los dispositivos de la arquitectura de un ordenador que sirve para almacenar la información, de tal forma que ésta puede leerse, escribirse, ejecutarse, o una combinación cualquiera de lo antes indicado. En las arquitecturas de los sistemas de producción pueden existir distintas memorias, conceptualmente distintas para la resolución de los problemas y el control del conocimiento.

Memoria de corto plazo. Memoria de trabajo. Parte de la memoria humana que se utiliza activamente cuando se piensa sobre un determinado problema. Por analogía con los ordenadores, la memoria de corto plazo es semejante a la memoria RAM, y contiene todos los términos conceptuados por trozos. Las últimas teorías congnitivas mantienen que los humanos disponemos de una memoria de este tipo que puede manipular hasta cuatro «trozos» o unidades cada vez.

Memoria de datos. Base de Datos global en un sistema de producción. El contenido puede ordenarse total o parcialmente en base a su tiempo de creación, o modificación más reciente. La memoria de datos suele ser la parte más volátil del sistema de producción.

Memoria de producción. Conjunto de todas las reglas de un sistema de producción. Los restantes componentes del sistema de producción son la memoria de datos y el motor de inferencias.

Memoria de reglas. Otra denominación de la memoria de producción.

Memoria de trabajo. Otro de los nombres de la memoria de datos.

Memoria izquierda. Estructura de datos de la red de algoritmos de contraste de Rete, que está asociada a un nodo. Contiene las combinaciones de los elementos de condición anteriores.

Metaconocimientos. En un sistema experto, conocimiento de cómo el sistema opera o razona, por ejemplo el conocimiento de cómo éste utiliza y controla el conocimiento del dominio. Conocimiento sobre el conocimiento.

Metarregla. Regla relacionada con los conocimientos de metanivel. Las metarreglas se utilizan para especificar las estrategias de resolución de conflictos, o para filtrar y ordenar reglas de dominio.

Método de inferencia. Técnica utilizada por el motor de inferencia para acceder a aplicar el conocimiento del dominio, por ejemplo los encadenamientos hacia delante y hacia atrás.

Métodos de solución de problemas. Procedimiento (algorítmico o heurístico) por el cual se busca la solución a un problema.

Métodos basados en la estructura. Métodos de programación que utilizan jerarquías estructurales para las relaciones procedurales y la herencia.

Métodos basados en reglas. Métodos de programación que realizan los encadenamientos hacia adelante y hacia atrás utilizando reglas SI-ENTONCES.

Métodos de base lógica. Métodos de programación que utilizan cálculos de predicados para estructurar el programa y guiar su ejecución.

Métodos orientados al acceso. Métodos de programación basados en el uso de «ensayos» que activan nuevos cálculos, cuando se cambian o se leen datos.

Métodos orientados al objeto. Métodos de programación basados en la utilización de elementos llamados objetos, que se comunican unos con otros a través de mensajes.

Métodos orientados al procedimiento. Métodos de programación que utilizan subrutinas anidadas para organizar y controlar la ejecución del programa.

Modelo cognitivo. Simulación del sistema cognitivo humano (entre otros elementos, percepción, habilidad para actuar, memoria y pensamiento) en términos de proceso de la información. Estos modelos suelen ser programas de ordenador.

Modelo de sistema de producción. Estilo de programación y resolución de problemas, caracterizado por una arquitectura de sistemas de producción. Otros modelos usuales son la programación procedural (por ejemplo, en PASCAL), programación simbólica (LISP), programación lógica (PROLOG) y programación orientada al objeto (SMALLTALK).

Modelos mentales (de expertos humanos). Redes simbólicas y patrones de relaciones que utilizan los expertos cuando intentan comprender un determinado problema. Los modelos mentales suelen tomar con fre-

cuencia la forma de analogías simplificadas, o metáforas, que utilizan los expertos la primera vez que examinan el problema. Estos modelos mentales pueden en ocasiones convertirse en reglas de producción, y son objeto de investigación.

Módulo de explicaciones. Parte de un sistema experto que explica cómo se alcanzan las soluciones y justifica los pasos utilizados para alcanzarlas.

Modus ponens. Regla de inferencia de la lógica: $A, A \rightarrow B \Rightarrow B$.

Monotónico. Unidireccional. En análisis, una función monotónica es aquella que o no se incrementa (sólo se mantiene y/o decrementa), o no se decrementa (sólo crece o se mantiene). En inferencias, una lógica monotónica sólo puede añadir proposiciones a una base de conocimientos, y no puede jamás eliminarlos.

Motor de inferencia. Parte del sistema basado en el conocimiento, o sistema experto que contiene el conocimiento para la resolución general del problema. El motor de inferencia procesa el conocimiento de un dominio (localizado en la base de conocimientos) para obtener nuevas conclusiones.

Mundos hipotéticos. Forma de estructurar el conocimiento, en un sistema basado en el conocimiento, que define los contextos (mundos hipotéticos) en los que son aplicables las reglas y los hechos.

Nodo de dos entradas. Nodos de la red de algoritmos de concordancia de Rete que intercala las concordancias para cada elemento de condición con los de todos los elementos de condición anteriores.

Nodo de producción. Nodo especial de la red de algoritmos de concordancia de Rete que asocia una regla con el conjunto de nodos que comprueban si las condiciones de dicha regla han sido satisfechas.

Nodo de una entrada. Nodo en la red del algoritmo de concordancia de Rete que está asociado con una prueba de un atributo único de un elemento de condición. Pasará un distintivo si, y sólo si, la prueba de atributo se satisface.

Objetivo. Final hacia el que tiende la resolución de un problema. En un sistema de producción, los objetivos se pueden representar en una memoria separada o en una clase de elementos de la memoria de trabajo, adecuadamente distinguible.

Objeto. Entidad de un sistema de programación, que se utiliza para representar el conocimiento declarativo, y, en ciertos casos, el conocimiento procedural sobre un objeto físico, un concepto o una estrategia de resolución de problemas.

Ordenación. Estrategia de resolución de conflictos en la cual la dominación de una instanciación sobre otra está determinada por un orden estático impuesto en las reglas.

Ordenadores de Quinta Generación. Forman la siguiente generación de ordenadores electrónicos. Se les supone más rápidos y amplios, y fundamentalmente con diseños nuevos. Con el aumento de la potencia, se espera que procesen varios programas simultáneamente (proceso paralelo). Como los sistemas expertos tienden a ser muy amplios y a necesitar mucho tiempo de procesamiento, se piensa que los sistemas expertos no alcanzarán la madurez hasta que se disponga de estas máquinas potentes.

PROLOG. Lenguaje de programación simbólica basado en el cálculo de predicados. El PROLOG es un lenguaje más popular en investigación sobre Inteligencia Artificial, fuera de los Estados Unidos.

Paquete de ruptura (Break package). Mecanismo de un lenguaje de ingeniería del conocimento o programación mediante el cual se indica al programa dónde debe detenerse para que el programador pueda examinar los valores de las variables.

Paradigma de consulta. Los paradigmas de consulta describen tipos de escenarios genéricos de resolución de problemas. Cada herramienta de construcción de sistemas es buena para algunos paradigmas de consulta, pero no para otros. La mayoría de las herramientas comercializadas han sido diseñadas para facilitar el desarrollo rápido de los sistemas expertos que trabajan con el paradigma diagnóstico/prescriptivo.

Paradigma de consulta de diagnóstico/prescriptivo. Los paradigmas de consulta se refieren a aproximaciones genéricas a tipos de problemas comunes. El paradigma de diagnóstico/prescriptivo se utiliza con problemas que requieren que el usuario identifique síntomas o características de una situación, para determinar cuál de las soluciones alternativas es la adecuada. La mayor parte de los sistemas expertos y herramientas están diseñados para que puedan manejar éste paradigma.

Paralelismo. Realización de varias operaciones en una unidad de tiempo. El hardware de un ordenador paralelo ejecuta más de una instrucción de máquina por cada ciclo. Los sistemas de producción paralelos disparan

más de una instanciación por cada ciclo de reconocimiento. En los ordenadores serie, el paralelismo puede siempre simularse (aunque sin alcanzar la velocidad de los ordenadores paralelos).

Patrón. Descripción abstracta de un dato que supone algunas restricciones al valor o valores que puede tomar, pero que no es necesario que se especifiquen con todo detalle.

Pila de reglas. Conjunto de reglas que funcionan juntas para obtener un objetivo, o bien el conjunto de reglas relacionadas con un elemento de contexto.

Pizarra. Base de datos accesible a fuentes de conocimiento independientes, y utilizada por ellas para su comunicación. La información incluye generalmente resultados intermedios de la resolución de problemas.

Poda. En sistemas expertos, proceso por el cual se cortan o ignoran una o más ramas de un árbol de decisión. En efecto, las reglas heurísticas pueden reducir el espacio de búsqueda, determinado de ciertas ramas (o subconjuntos de reglas) se pueden ignorar.

Postcondición. Proposición que debe ser satisfecha antes de la ejecución de un trozo de código determinado. Si la precondición se satisface y el código se ejecuta correctamente, la postcondición será verdadera en cuanto se ejecute el código.

Problema de mundo real. Problema práctico y complejo que tiene una solución útil en el sentido de mejora de la relación coste-eficacia.

Procedimiento. Conjunto de instrucciones para realizar una tarea. También puede ser un programa que lleva un algoritmo o heurístico. Asimismo, puede ser una unidad sintáctica de un programa en lenguaje procedural, que puede ser parametrizada, de forma que se pueda invocar el mismo segmento de código desde diferentes puntos del programa, con datos diferentes. Finalmente, también puede ser cualquier representación de conocimientos procedurales.

Procedural versus declarativo. Dos formas complementarias de considerar un programa ordenador. Los procedimientos indican al sistema qué debe hacer (por ejemplo multiplicar A por B y luego añadirle C). Las declaraciones indican al sistema qué debe saber.

Proceso de Información Humana. Es una perspectiva de cómo piensan las personas, influenciada por cómo funcionan los ordenadores. Este

sistema psicológico comienza centrándose en la información que la persona utiliza para alcanzar alguna conclusión, y a continuación, se pregunta cómo se podría diseñar un programa de ordenador que comience con esta misma información para terminar alcanzando la misma conclusión. Esta perspectiva ha sido desarrollada por Herbert Simon y Alan Newel, y domina la psicología cognitiva, influenciando el diseño de lenguajes y programas de ordenador.

Producción. Término utilizado por la filosofía cognoscitiva para describir una regla SI-ENTONCES.

Producto cartesiano. En teoría de conjuntos, el producto cartesiano de un conjunto A por otro conjunto B es el conjunto formado por todos los pares ordenados (a,b) de tal modo que a pertenezca al conjunto A, y b al conjunto B. En una discusión del proceso de concordancia, el término se utiliza para designar el conjunto de todas las combinaciones potenciales de concordancias para una secuencia de elementos de memoria de trabajo.

Programa basado en reglas. Programa similar en cierto modo a un programa de sistema de producción en que el conocimiento se representa explícitamente mediante reglas en lugar de hacerlo mediante reglas en lugar de hacerlo mediante procedimientos, pero los programas basados en reglas no están implementados necesariamente en los lenguajes de sistemas de producción de propósito general, y no necesitan hacer uso exclusivo de la arquitectura de los sistemas de produccion.

Programa de un sistema de producción. Programa de aplicación escrito en un lenguaje de sistema de producción y en el que la mayor parte del problema se lleva a cabo disparando las reglas.

Programación automática. Se están desarrollando algunos proyectos de programas de ordenador, que a su vez sean capaces de escribir otros programas de ordenador. Si se alcanzan los objetivos propuestos, estos programas de alto nivel se utilizarán para automatizar una gran parte de los programas.

Programación simbólica o numérica. Los sistemas del conocimiento utilizan la programación simbólica para manipular las cadenas de símbolos lógicos, no operadores numéricos. Son pues dos formas primarias de utilizar los ordenadores.

Propagación de probabilidades. En una red de inferencias, ajuste de

probabilidades en los nodos, para tener en cuenta el efecto de una nueva información sobre la probabilidad en un nodo particular.

Propiedad. Una de las características de un objeto. Las propiedades que tienen valores se llaman atributos. Los componentes de las estructuras y esquemas que pueden ser de complejidad arbitraria se llaman en ocasiones propiedades.

Prototipo. En desarrollo de sistemas expertos, un prototipo es una versión inicial del sistema experto, con una 25 a 200 reglas, y que se desarrolla para obtener más eficacia de la representación global del conocimiento y de las estrategias de inferencia para resolver cada problema específico.

Prueba de teoremas por resolución. Es una de las utilizaciones específicas de la lógica deductiva para probar teoremas en el cálculo de predicados de primer orden. El método utiliza el siguiente principio de resolución: (Av B) y (–A v C) implica (B v C).

Ranura. Componente de un objeto en un sistema de marcos. Las ranuras pueden contener características intrínsecas como por ejemplo el nombre del objeto, atributos y valores, atributos con valores por omisión, indicadores que aludan a otras estructuras relacionadas, e información sobre el creador de la estructura, etc.

Razonamiento. Proceso por el cual se trazan inferencias o conclusiones.

Razonamiento dirigido mediante supuestos. Estrategias de resolución de problemas que genera hipótesis sobre los acontecimientos que se esperan ocurran, centrándose en procesar las tareas relacionadas con esos acontecimientos. La actividad de esta estrategia contrasta con la pasividad del razonamiento dirigido a los datos.

Razonamiento no monotónico. Técnica de razonamiento que soporta múltiples líneas de razonamiento (varios caminos para alcnzar la misma conclusión) y en la que se admite la posibilidad de asignar nuevos valores a hechos o conclusiones aparte de una nueva información obtenida. Es útil para procesar conocimientos y datos que no ofrecen confianza.

Recency. Término inglés con el que se designa el método de eliminación por orden temporal. Estrategia en la resolución de conflictos que favorece las instanciaciones con aquellos elementos de la memoria de trabajo que fueron los últimos creados o modificados.

Red de activación. Una red de activación es un gráfico, en el que cada nodo representa un objeto cada uno de cuyos arcos representa una relación entre dos objetos. Si el arco lleva etiqueta, ésta consistirá en un número que indica la fuerza de esta relación. Cuando se procesa uno de estos nodos, su nivel de activación puede cambiar, y los efectos de este cambio se propagan por los arcos a los nodos relacionados, resultando de ello cambios en su nivel de activación.

Red de inferencias. Todas las posibles cadenas que pueden ser generadas por las reglas, en un sistema basado en reglas.

Red semántica. Estructura de datos para representar el conocimiento declarativo. Esta estructura es un gráfico en el que los nodos representan conceptos (elementos diversos), y los arcos (que pueden llevar etiquetas) representan las relaciones entre los conceptos.

Redundancia temporal. Tendencia de los sistemas de producción a realizar pocos cambios en la memoria de datos (y por tanto en el conjunto conflicto) en el espacio de tiempo que va desde un ciclo de reconocimiento al siguiente.

Refinamiento por pasos. Metodología de programación en la que el programa se especifica de forma abstracta y, mas tarde, en pasos sucesivos, las distintas partes del programa se sustituyen por instanciaciones más concretas de las descripciones abstractas. Este término es casi sinónimo de programación de arriba a abajo.

Reformulación del problema. Conversión de un problema organizado de forma arbitraria en otra forma que lleva directamente a una solución rápida y eficaz.

Refracción. Estrategia en la resolución de conflictos que evita que se dispare una instanciación, si ya ha sido disparada en otra ocasión anterior. La refracción toma su nombre de una propiedad específica de las neuronas. Durante el período de refracción (durante un corto intervalo posterior a su disparo) las neuronas no se disparan con los estímulos.

Regla. Modelo formal utilizado para especificar una recomendación, directiva o estrategia, expresada en forma de «premisa, conclusión» o bien de «condición, acción» (SI premisa, ENTONCES conclusión, o SI condición, ENTONCES acción).

Regla de producción. Es un tipo de regla que se utiliza en un sistema de producción. Generalmente, se expresa como SI condición ENTONCES acción.

Reglas heurísticas. Reglas escritas para expresar los procedimientos heurísticos que utiliza un experto para resolver el problema. Puede que el procedimiento heurístico original no se haya transformado en reglas «si-entonces», y por tanto uno de los problemas que hay que abordar en la construcción de un sistema del conocimiento es convertir un conocimiento heurístico en reglas. La potencia de un sistema basado en el conocimiento viene dado por las reglas heurísticas existentes en la base de conocimiento.

Reglas si-entonces. Instrucción de relación entre un conjunto de hechos. Estas relaciones pueden ser definitorias (por ejemplo, si mujer casada, entonces esposa) o heurísticas (por ejemplo, si está lluvioso, entonces coje el paraguas).

Relaciones consistentes. Conjunto de valores que se relacionan con las variables, que satisface las condiciones de cada patrón, tomado uno a uno, y también satisface todas las limitaciones aplicables al conjunto de patrones.

Representación. Forma en la que el sistema almacena el conocimiento sobre un determinado dominio. El conocimiento está formado por hechos, y relaciones entre esos hechos.

Representación del conocimiento. Proceso de estructuración del conocimiento que se tiene acerca de un problema, de forma que se facilite la resolución de dicho problema.

Resolución. Estrategia de inferencia utilizada en sistemas lógicos para determinar la verdad de una determinada aserción. Este método, muy complejo pero efectivo, establece la verdad de una aserción, determinando si se encuentran o no contradicciones cuando se intenta resolver las cláusulas, una de las cuales es la negación de la tesis que se desea afirmar.

Resolución de conflictos. Técnica de resolución del problema de concordancias múltiples, en un sistema basado en reglas. Cuando más de un antecedente de una regla concuerda con la base de datos, se produce un conflicto, ya que a) según lo establecido cada regla que concuerde podría ser ejecutada a continuación, y b) sólo se puede ejecutar a continuación una regla. Un método de resolución de conflictos muy común consiste en asignar a cada regla una prioridad, y, de este modo, la regla cuya prioridad sea más alta (entre las que concuerda) será la que se ejecute a continuación.

Robótica. Rama de la Inteligencia Artificial que está relacionada con

«hacer que los ordenadores vean o manipulen objetos de su entorno». La inteligencia artificial no es la robótica, pero está relacionada con ella en el sentido que desarrolla las técnicas necesarias para la construcción de robots que utilizan métodos heurísticos para funcionar de forma flexible, interactuando con un entorno cambiante constantemente.

Robustez. Una de las cualidades de un sistema de resolución de problemas, que permite la degradación gradual de las características, si se lleva a los límites de sus posibilidades de expertización o si se le dan datos o reglas erróneos, inconsistente o incompletos.

Satisfacer. Estrategia de resolución de problemas que tiene éxito cuando la solución potencial satisface un criterio mínimo de aceptabilidad especificado. La solución no es necesariamente la óptima. En muchos problemas, la búsqueda de la solución óptima es innecesaria, y puede llegar a ser un derroche de tiempo prohibitivo.

Sección de declaración. Parte de un programa de ordenador en la cual construcciones tales como tipos de datos, variables, procedimientos y funciones están enunciadas y, en algunos casos, definidas. Es usual que la sección de declaración defina los elementos de la memoria de trabajo, índices de los atributos, atributos de vector, funciones externas, etc.

Semántica. Que se refiere al significado, en contraposición a sintáctico, o perteneciente a la forma.

Semántica dual. Un programa de ordenador puede ser considerado desde una cualquiera de las dos siguientes perspectivas: semántica procedural (que tiene lugar cuando el programa se ejecuta) y semántica declarativa (que indica qué conocimiento contiene el programa).

Sensitividad. Respuesta de un sistema a las demandas cambiantes de su entorno. La sensitividad de un sistema de producción se ve influenciada por la estrategia de resolución de conflictos.

Símbolo. Signo arbitrario que se utiliza para representar objetos, conceptos, operaciones, relaciones o cualidades.

Similitud estructural. En un sistema de producción, elementos sintácticos comunes existentes en las reglas. Las reglas que tienen los mismos valores para los atributos, elementos de condición idénticos o secuencias de idéntica condición, son estructuralmente similares. El algoritmo de concordancia de Rete, aprovecha la similitud estructural para compartir cadenas de nodos con una entrada, en sus redes, para aquellas redes que tengan secuencias de condiciones idénticas.

Sintáctico. Perteneciente a la forma o estructura. En contraposición con semántico, o perteneciente al significado.

Sistema «runtime» o versión «runtime». Las herramientas de construcción de los sistemas basados en el conocimiento permiten al usuario crear y ejecutar varias bases de conocimiento. Con una única herramienta, el usuario puede crear docenas de bases de conocimiento. Dependiendo del problema al que se encare el usuario, cargarán una base de conocimientos apropiada y llevarán a cabo consultas. Con esta herramienta, el usuario puede modificar fácilmente una base de conocimientos. Las compañías que desarrollan bases de conocimiento específicas no desean que el usuario «cargue» la base de conocimientos, ni que sea capaz de modificarla. Cuando una herramienta de construcción de un sistema experto se modifica para incorporar una base de conocimientos específica, y para desactivar algunas de las características de programación, el sistema resultante se dice que es un «runtime system» o una versión «runtime». (A veces se llaman sistemas de «sólo ejecución»).

Sistema basado en el conocimiento. Programa en el cual el conocimiento del dominio es explícito y separado de otros conocimientos del programa.

Sistema conducido por ejemplos. Véase Sistema por Inducción.

Sistema de inducción. Sistema basado en el conocimiento que tiene una base de conocimientos formada por ejemplos. A partir de estos ejemplos, un algoritmo de inducción construye un árbol de decisiones, y el sistema pide consejo. Los sistemas de inducción no facilitan el desarrollo de jerarquías de reglas.

Sistema de producción controlado. Sistemas de producción en el que el régimen de control se especifica mediante una máquina de estados finitos para cada conjunto de reglas, en lugar de por la arquitectura del sistema de producción.

Sistema de producción de Post. Modelo matemático de cálculo diseñado por Emil Post, que se considera origen de todos los demás sistemas de producción.

Sistema de producción. Arquitectura para la resolución de problemas, que emplea un conjunto de reglas (almacenadas en la memoria de producción), una base de datos global (almacenada en la memoria de datos) y un motor de inferencia que realiza el ciclo de reconocimiento de concordancias, resolución de conflictos y disparo de una regla.

Sistema experto. Programa de ordenador (escrito con frecuencia en un lenguaje de sistemas de producción), que es experto en un dominio concreto.

Sistemas basados en el conocimiento pequeños. En general, son sistemas pequeños que contienen unas 500 reglas. Se han diseñado para ayudar a resolver dificultades de análisis y de toma de decisiones, sin aspirar a ser equivalentes a los expertos (humanos).

Sistemas basados en reglas. Es un programa organizado como un conjunto. Véase reglas.

Situación. Estado de la memoria de datos que corresponde a algún conjunto de propiedades del dominio que se están modelando por el sistema de producción. En ocasiones se designa a las reglas como parejas situación-acción.

Software, niveles de. Gama continua que comienza, en el nivel inferior, con el lenguaje máquina y que se extiende a software de bajo nivel, de alto nivel, herramientas, y finalmente a sistemas que los usuarios pueden utilizar para resolver problemas.

Subobjetivo. Objetivo que cuando se satiface es suficiente para asegurar que se ha obtenido también otro objetivo. En sistemas con encadenamiento hacia atrás, los objetivos sin alcanzar se descomponen en subobjetivos más simples, con la esperanza de que éstos se puedan resolver con mayor facilidad. En los árboles de objetivos, la relación de un objetivo con sus subobjetivos está representada como una relación padre-hijo.

Subproblema. Otra de las denominaciones de subobjetivo.

Suposición. Aseveración que no se conoce o de la que no se tiene la certeza de su veracidad, en contraposición con hecho. También puede referirse a la confianza en la validez de una aseveración de la que no se conoce o supone su veracidad. El grado de confianza puede cuantificarse mediante un cierto factor de certeza o factor de confianza.

Tarea. Otro nombre para designar un contexto, o en sistemas de encadenamiento hacia atrás, objetivo.

Ternas Objeto-Atributo-Valor (O-A-V). Método de representación del conocimiento factual. Es el método utilizado en EMYCIN. Un objeto es una entidad real o conceptual dentro del dominio del consultante (por ejemplo, un paciente, una compañía de aviación, etc.). Los parámetros son

propiedades asociadas a cada objeto (por ejemplo, edad y sexo de un paciente, situación económica y localidad donde se encuentra la sede de la compañía de aviación, etc.). Los atributos pueden tomar valores, como por ejemplo, «29 años».

Transferencia de tecnología. En el contexto de los sistemas expertos, es el proceso por el cual los ingenieros del conocimiento adecúan un sistema experto para un grupo de usuarios. Como los sistemas expertos necesitan estar actualizándose continuamente, los ingenieros del conocimiento necesitan «entrenar» a los usuarios a mantener el sistema, antes de recibirlo en su entorno. Por tanto, los usuarios necesitan aprender cierta ingeniería del conocimiento.

Up-down. Estrategia de proceso. Ver de arriba a abajo.

VLSI. Siglas de Very Large Scale Integration. Integración a gran escala. Sistema de construcción de circuitos electrónicos muy densos y potentes, cuyo tamaño es el de una pequeña oblea. Esta tecnología ha resultado fundamental para el diseño de ordenadores grandes y potentes, cuyo volumen, por el contrario, es pequeño. Existen sistemas expertos para el diseño de este tipo de circuitos.

Valor. Cantidad o cualidad que se puede utilizar para asignar a un atributo. Si consideramos, por ejemplo, el atributo «color», los posibles atributos de color son todos los nombres de colores que conocemos. Si consideramos un objeto en particular, estaremos asignando un valor específico al atributo en cuanto indiquemos «El coche es rojo brillante», por ejemplo.

Valores activos. Valores especiales que pueden cambiar fácilmente en el curso de una consulta. Los valores activos se utilizan con frecuencia con imágenes gráficas para permitir al usuario cambiar los valores de un sistema sin más que alterar una imagen de la pantalla del ordenador.

Valores por defecto. Los programas de ordenador suelen utilizar ciertos valores preespecificados que utilizan, si no toman otros valores alternativos. A estos valores «supuestos» se les denomina valores por defecto. Los sistemas del conocimiento suelen almacenar valores por defecto, para utilizar en lugar de hechos. Como ejemplo, citemos un programa médico que supone que un paciente ha estado expuesto a la acción de un organismo determinado, si el usuario no indica que dicha exposición queda descartada.

Variable. Término dentro de una estructura de datos o de proceso que

puede tomar un valor cualquiera, dentro de una gama de valores. Estos valores pasan a ser el Dominio de la variable, que puede determinarse por sus propiedades sintácticas o semánticas. En matemáticas, las variables se utilizan para permitir aplicar todos los valores a las proposiciones del dominio. En programación, a las variables se les puede asignar más de un valor en cada momento (considerando el tipo).

Variable de elemento. Variable que está ligada a un elemento de la memoria de trabajo completo, en lugar de estarlo a los valores de los atributos de ese elemento.

Variable de segmento. Variable de un patrón que concuerda con una subparte de una lista.

Vector de estado. Lista ordenada que describe completamente el estado del sistema. El vector está formado por un número de elementos fijo, y cada elemento es un parámetro del estado actual.

Ventanas. Los ordenadores y terminales convencionales utilizan la pantalla completa para presentar la información. Los ordenadores que disponen de software adecuado, utilizan varias ventanas (zonas o divisiones de la pantalla) para presentar la información, que se toma de distintas bases de datos. Así, por ejemplo, se puede tener un procesador de texto funcionando en una ventana, y un programa gráfico, en otra. La mayor parte de la investigación sobre sistemas expertos se realiza en ordenadores que disponen de varias ventanas. Esta técnica ha sido desarrollada por investigadores en Inteligencia Artificial.

Workstation. Ver Estación de trabajo.

Bibliografía

• J. L. Alty y M. J. Coombs. *Sistemas expertos: Conceptos y ejemplos.* Ed. Díaz de Santos. 1986.

• J. P. Aubert y R. Schomberg. *Inteligencia Artificial.* Ed. Paraninfo. 1986.

• M. A. Bramer. *Research and Development in Expert Systems.* Ed. Cambridge University Press. 1984.

• Eugene Charniak y Christopher K. Riesbeck y McDermott, V. Drew. *Artificial Intelligence Programming.* Ed. Lawrence Erlbaum Associates, Publishers. 1980.

• Dimitris Chorafas, N. *Applying Expert Systems in Business.* Ed. McGraw-Hill Book Company. 1987.

• Paul R. Cohen y Edward A. Feigenbaum. *The Handbook of Artificial Intelligence.* (3 tomos). Ed. Heuristech Press y William Kaufmann. 1982.

• J. P. Delahaye. *Systèmes Experts: organisation et programmation des bases de connaissance en calcul propositionel* Ed. Eyrolles. 1987.

• Michel Gondran. *Introduction aux Systèmes Experts.* Ed. Eyrolles. 1986.

• Michel Grudnstein y Patrick de Bonniers. *De la fertilisation à l'insertion.* Revue de Intelligence Artificielle. 1987 (vol. 1, número 2).

• Paul Harmon y David King. *Expert Systems. Artificial Intelligence in Business.* Ed. Wiley Press. 1985.

• Chris Naylor. *Construya su propio Sistema Experto.* Ed. Díaz de Santos, S. A. 1984.

• G. L. Simons. *Expert Systems and Micros.* Ed. NCC Publications. 1985.

• Donald A. Waterman. A Guide to Expert Systems. Ed. Addison-Wesley Publishing Company. 1986.

Después de situar los «sistemas expertos» en el contexto de la Inteligencia Artificial y describir su construcción, su funcionamiento, su utilidad, etc., se analiza el papel que pueden tener en el futuro (y en el presente, ya) de la Informática, así como los polémicos temas de la «capacidad para desbancar a la inteligencia humana», y las posibilidades de «aprender» de que se puede dotar a un procesador, etcétera.